

Managing Signon & Password Access • Intelligent Graphic Interfaces • PM Programming Primer

September/October 1993  
Display Until 10/31/93

# OS/2 Developer

THE MAGAZINE FOR ADVANCED SOFTWARE DEVELOPMENT

**Spotlight on  
Digitalk**

**Remote  
LAN Access**

**Netware  
for OS/2**

**More on  
Controls**

**Performance  
Fine-Tuning**

**Transparent  
Networking  
Under  
OS/2**

U.S. \$9.



74470 12531



# WATCOM™

## Visual Development Environment For

**WATCOM VX•REXX** is an easy to use visual development environment for creating applications that leverage the capabilities of OS/2 2.x and exploit the Presentation Manager graphical user interface. VX•REXX combines a project management facility, visual designer and an interactive source-level debugger to deliver a very approachable and highly productive visual development environment.

### Design Applications

**Visually** Create rich graphical applications quickly and easily using the visual design environment. With the visual designer, you can graphically create Presentation Manager interface objects, quickly customize their properties, and easily attach REXX procedures to the objects.

### Integrated Development Environment

Build, test and debug your application without leaving the development environment. Then package your application as an EXE file or PM macro for royalty-free redistribution. The power of the integrated development environment and debugger can also be used with your existing REXX applications.

### Powerful Open Environment

Enjoy the simplicity of event-driven programming together with the global editing capabilities essential for professional project management. WATCOM VX•REXX is open and extensible through IBM's object oriented System Object Model (SOM) technology. You can access all standard REXX API's including DB Manager, because VX•REXX is based on the OS/2 2.x standard system REXX.

### Highlights

- ▶ Easy to use visual development environment
- ▶ Create and modify objects dynamically at both edit and run time
- ▶ Powerful project management facility
- ▶ Advanced interactive source-level debugger
- ▶ Package your applications as EXE files or PM macros
- ▶ Access to standard REXX API's including DB Manager
- ▶ System Object Model (SOM) based object manager
- ▶ Support for multi-threaded applications
- ▶ Include OS/2 style help and hints in your applications
- ▶ Supports SAA CUA'91 objects
- ▶ Autosizing and alignment of objects
- ▶ Integrated console window support for existing REXX programs
- ▶ Royalty-free run-time available
- ▶ Multiple modeless window support
- ▶ Create PM macros for applications supporting REXX as a macro language





# VX•REXX™

## OS/2 REXX

### Interactive Debugging

If an error occurs at run-time, VX•REXX will display a traceback pinpointing the source line where the error occurred. A simple click of the mouse will return you to the source edit window to correct the error. The built-in interactive source-level debugger lets you set breakpoints, step through code and watch variables to track down complex problems.

### Build Professional Applications

WATCOM VX•REXX allows you to leverage key OS/2 features to create professional applications. Build applications that dynamically create and modify CUA'91 screen objects at both edit and run-time, and include OS/2 style help and hints.

### Create Multi-Threaded Applications

Every VX•REXX application contains multiple threads. One thread remains responsive to user input while others continue processing. In addition, VX•REXX provides the ability for advanced applications to easily use additional threads.

Suggested Retail: \$299\*

**Introductory Offer \$99\***

(until September 30, 1993); includes royalty-free runtime

**Call Toll Free**

**1-800-265-4555**

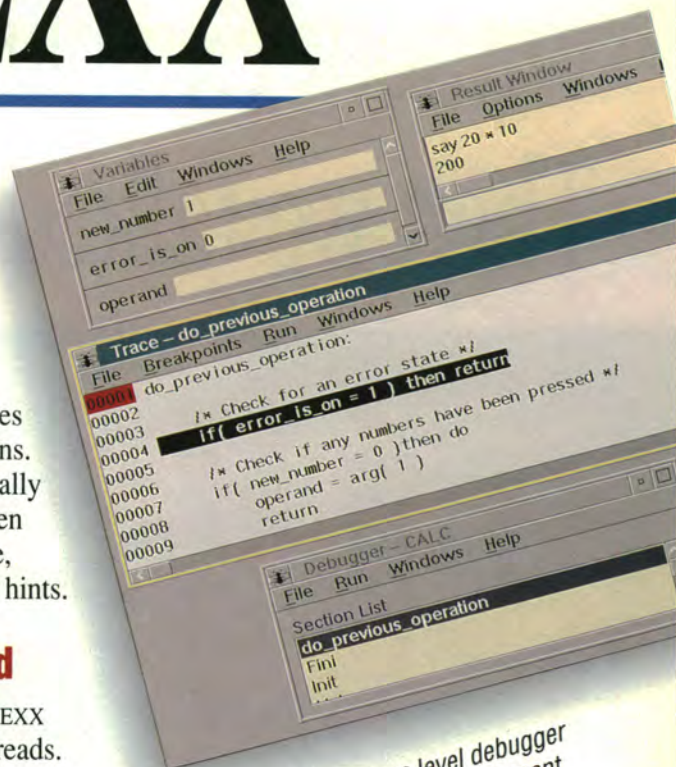
**WATCOM**

**WATCOM International**

415 Phillip Street, Waterloo, Ontario, Canada, N2L 3X2

Phone: (519) 886-3700 Fax: (519) 747-4971

\*Prices and specification are subject to change without notice. Price does not include freight and taxes where applicable. Prices quoted in US dollars. WATCOM, the Lightning Device, and VX•REXX are trademarks of WATCOM International Corporation. Other trademarks are the properties of their respective owners. © Copyright 1993 WATCOM International Corporation.



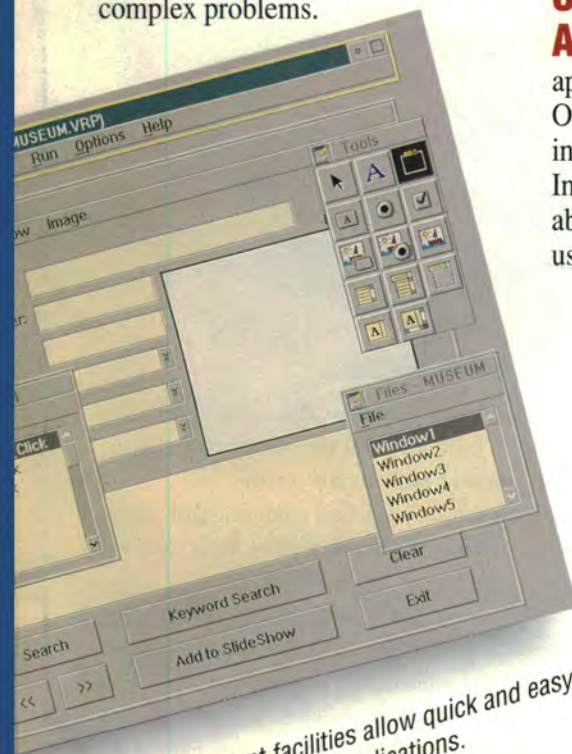
The integrated source level debugger simplifies your project development.



**WATCOM VX•REXX:**

The integrated visual solution builder for OS/2 2.x.

Project management facilities allow quick and easy development of your OS/2 applications.





“We were told it was impossible to develop a client/server application without extensive retraining. Then we talked to Micro Focus.”



Larry Lowder, Systems Architect, Questar Service Corporation.

Mountain Fuel Supply®, a division of Questar®, is a utility company supplying natural gas to 750,000 customers across Utah, Wyoming, Idaho and Colorado. The company's success is largely driven by its implicit belief that the customer is number one.

Yet, IT also plays its part in that success: client/server architectures and graphical user interfaces (GUIs) have helped Mountain Fuel Supply move applications and information closer to the customers and the employees. All of which has resulted in an augmented level of service being offered to customers.

When Larry Lowder, one of Questar's Systems Architects, set out to build the client/server architecture for Mountain Fuel Supply, he needed solutions, not skepticism. For the first project, a cashing system, he needed to link workstations with OS/2® to the DB2® database on the host, running CICS®.

"We were faced with having to spend up to two years retraining our

COBOL programmers in C and API calls. Then we discovered Micro Focus Dialog System™. It allowed us to build the client functionality we required, and re-engineer the existing mainframe application as a server."

"Within a week, mainframe programmers were producing GUI screens for COBOL. Within 90 days we had delivered the system. Now we're not only coming in under budget, but also way ahead of schedule."

As Mountain Fuel Supply discovered, the Micro Focus solution lets you make a productive transition to client/server without sacrificing any of the resources you've worked so hard to build.

When the world's leading corporations demand "A Better Way of Programming,"™ they turn to Micro Focus. For a brochure on putting the Micro Focus Client/Server Solution to work for you, call 800-872-6265.

**MICRO FOCUS**®

Micro Focus Inc. 2465 East Bayshore Road, Palo Alto, CA 94303. Tel. (415) 856 4161.



# OS/2 Developer

SEPTEMBER/OCTOBER 1993

THE MAGAZINE FOR ADVANCED SOFTWARE DEVELOPMENT



## EDITOR'S COMMENTS

Two Paths Converge 5



## SPOTLIGHT

Digitalk: Objects Pioneer Pushes the Envelope 6



## LAN

Network SignON Coordinator/2: A Plus For Network Applications 19

The LAN Distance Product: LAN Application Transparency for the Remote User 29

Using NetWare 4.01 for OS/2 37



## GUI

A Color-Full Example: Using Color In Control Design 46

Graphics Interface Kit/2: Visualizing and Manipulating Data 59

Demystifying the OS/2 Presentation Manager 70



## MULTIMEDIA

Programming the Graphical User Interface Extensions of MMPPM/2 80



## 32-BIT

Converting Real Addresses to Virtual Addresses in OS/2 2.x 86



## PRODUCT WATCH

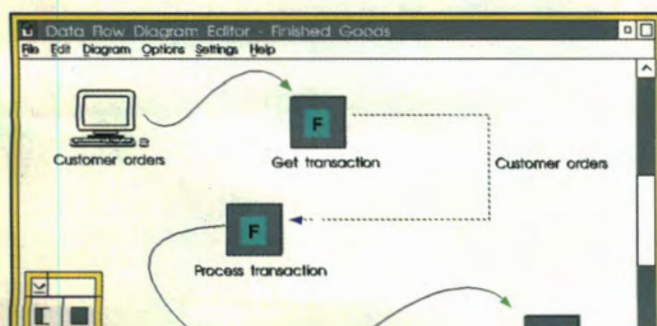
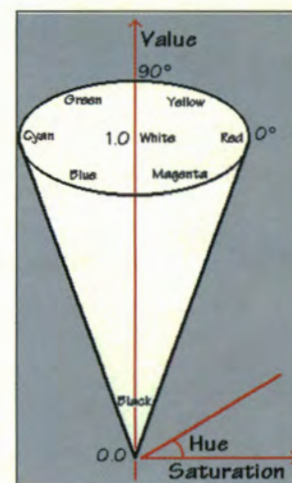
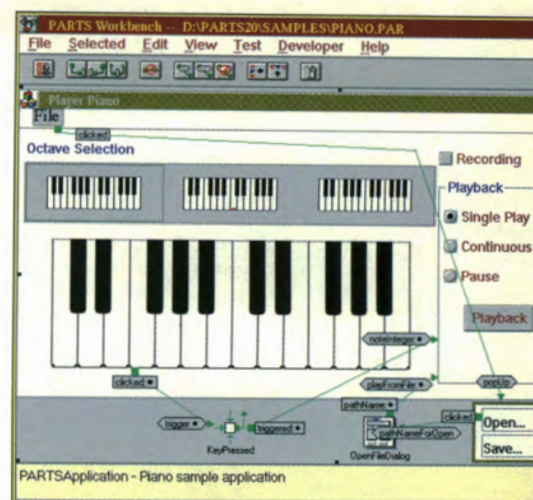
More REXX, GUI, SOM, And Other Tools 97



## PERFORMANCE

SPM/2: Fine-Tuning Your Application 100

Using LINK386 Options To Improve Performance 109







# Programmer's Paradise®

Programmer's Paradise®  
Italia  
Phone: 39-2-480-16053  
FAX: 39-2-480-16039

## IBM® OS/2® 2.1 by IBM Corporation

OS/2's pre-emptive, multithreaded multitasking allows you to run multiple tools concurrently. This means you can edit in one session while a compile-and-link occurs in another, and a print job occurs in a third. By operating pre-emptively, OS/2 makes sure all running applications get the time they need to run, so that overall performance is improved.



List: \$249 Ours: \$139  
Upgrade List: \$199 Ours: \$ 99  
FAXcetera #: 3142-0009

## BocaSoft Wipeout & BocaSoft System by BocaSoft Inc.



**BocaSoft WipeOut** is a 32-bit screen saver for OS/2 featuring numerous animated displays integrated with digital audio, password protection, screen capture, randomizer, and on-line help. A developer kit is included that allows the development of new screen savers.

**BocaSoft System Sounds** is a 32-bit OS/2 application that allows the association of system events, keystrokes and mouse events with digital audio.

WipeOut List: \$59 Ours: \$55  
System Sounds List: \$59 Ours: \$55  
FAXcetera #: 1010-2501

## WATCOM VX•REXX by WATCOM Int'l Corp.

WATCOM VX•REXX is an easy to use visual development environment for creating applications that leverage the capabilities of OS/2 2.x and exploit the Presentation Manager graphical user interface. VX•REXX combines a project management facility, visual designer and an interactive source-level debugger to deliver a very approachable and highly productive visual development environment.



List: \$299 Ours: \$99  
Special Introductory Offer! (until Sept. 30, 1993)  
FAXcetera #: 1683-0016

## SQL Objects++™ Database Class Library by Graphical Software Interfaces, Inc.



SQL Objects++™ Database Class Library is a powerful collection of object-oriented database classes that provide direct access to SQL and non-SQL databases. Each class is fully optimized to provide fast access to your data. No other access library is designed to fully utilize your OS/2 environment.

List: \$499 Ours: \$249  
Special Introductory Offer! (until Oct. 31, 1993)  
FAXcetera #: 1010-2601

## Greenleaf Comm++ v2.0 by Greenleaf Software, Inc.

Comm++ will accommodate interrupt-driven circular buffered service for 35 ports at baud rates to 115,200. As a C++ communications library, it provides a hierarchy of classes which give the programmer simple access and control of serial communications. Classes are provided for: serial port controls, modem controls, file transfer protocols and calculation of check values.



List: \$249 Ours: \$199  
FAXcetera #: 1035-0011

## Error Manager™ by Soft & GUI, Inc.

Introducing Error Manager (EM), the runtime API debugger and production code monitor. EM finds elusive API errors and enables Testers and Production Managers to monitor software in real user environments. Best of all, EM works with fully optimized code and does not require symbol information or the Debug Kernel—yet it will tell you which line in your source is causing the error!



List: \$225 Ours: \$169  
FAXcetera #: 1002-0301

## C++/Views for OS/2 by Liant Software Corporation

Use C++/Views to help you quickly create high-quality 32-bit GUI applications for OS/2 2.x! What's more, applications developed with C++/Views are fully source-code portable to the other major operating environments—MS Windows, Motif, and Macintosh. C++/Views achieves this portability without sacrificing features, performance, or a native look and feel.



List: \$999 Ours: Call for special pricing  
FAXcetera #: 1952-0021

## GUARANTEED BEST PRICES! (Call for details)

To order call: 800-445-7899  
Corporate (CORSOFT): 800 422-6507  
FAX: 908 389-9227  
International: 908 389-9228  
Customer Service: 908 389-9229  
For more information on the products  
featured on these pages call—  
FAXcetera®: (201) 762-1975



Programmer's Paradise®  
1163 Shrewsbury Avenue  
Shrewsbury, NJ 07702

• All prices are subject to change without notice.

800-445-7899



**EDITOR**

Dick Conklin

**EXECUTIVE EDITOR**

Nicole Freeman

**PRODUCTION EDITOR**

Peter Altenberg

**COPY/PRODUCTION EDITOR**

Lisa Gluskin

**GROUP DIRECTOR**

Don Pazour

**PUBLISHER**

Cathy Passage

**ADVERTISING SALES**

Jo Ben-Atar

(203) 498-0329

Michele Blake

(212) 626-2322

Dave Moreau

(212) 626-2318

**MARKETING MANAGER**

Susan McDonald

**ART DIRECTOR/MARKETING**

Christopher H. Clarke

**ADVERTISING PRODUCTION****COORDINATOR**

Tom Marzella

**DIRECTOR OF PRODUCTION**

Andy Mickus

**DIRECTOR OF CIRCULATION**

Jerry Okabe

**CIRCULATION DIRECTOR**

Moir Boyle

**SUBSCRIPTION INQUIRIES**

U.S.: (800) WANT-OS2

International:

(708) 647-5960



BY DICK CONKLIN

# Editor's Comments

## Two Paths Converge

**T**his magazine is now beginning its sixth year of publication—as always, propelled by change. If you can pardon a bit of nostalgia, I'd like to bring you—especially our newer readers—up to date on where we've been and where we're going.

**Communicate!** IBM Personal Systems Developer was launched in 1988, following a survey of independent software vendors enrolled in IBM's Developer Assistance Program. DAP members had asked IBM to improve its communications with them; a "newsletter" was suggested. I had just written *OS/2: A Business Perspective* and saw the need for a regular OS/2 publication. The newsletter became a magazine, and the first issue (a skinny 48 pages) just barely made COMDEX Fall '88, where we handed out 25,000 copies.

**Big Blue, Too.** Although our target audience was software vendors, we soon noticed an increase in subscribers from IBM's traditional customer base. Corporate in-house programming departments were also adopting OS/2 and shared the need for current technical advice. Both groups want to optimize performance, exploit the Workplace Shell, write bug-free code, and move into the world of objects.

**Ready for Prime Time.** Last year we changed publishers, selecting Miller Freeman Inc., a San Francisco-based company with impressive credentials. MFI, which also publishes *Software Development*, *Dr. Dobbs Journal*, and *Microsoft Systems Journal*, knows the trade magazine business well and helped the renamed OS/2 Developer enter the retail market, find new subscribers, and start publishing bimonthly. Today we are an independent

OS/2 publication designed to serve commercial and corporate OS/2 developers worldwide.

**Five Years Old and Counting.** As we enter our second half-decade, it seems appropriate to remember some of the people who got us here: Bob Tabke and Debbie Dawson of the TDA Group, Jo-Ann Campbell of MLE Design, IBM's Lee Reiswig and former IBMers Don Barovich, Brian Proffitt, and Mike Engelberg. And of course the pros at MFI: Don Pazour, Regina Ridley, Cathy Passage, Nicole Freeman, Lisa Gluskin, and everyone in advertising, circulation, and production.



Dick Conklin

| Contact                                  | CompuServe                    | Internet                      |
|------------------------------------------|-------------------------------|-------------------------------|
| Dick Conklin<br>(Editorial)              | 76711,1005 or<br>OS2DF2 Forum | os2mag<br>@vnet.ibm.com       |
| Miller Freeman<br>(Circulation)          | 71572,341                     | 71572.341<br>@compuserve.com  |
| Cathy Passage<br>(Publisher)             | 71673,1163                    | 71673,1163<br>@compuserve.com |
| Lisa Gluskin<br>(Production/copy editor) | 75040,2553                    | 75040.2553<br>@compuserve.com |

This magazine was started because OS/2 developers asked for it. We'll continue to pursue the kind of articles that will help you develop leading-edge OS/2 applications. To do that, we need to hear from you. Keep those cards and letters coming—via CompuServe, Internet, fax, or mail.

Dick Conklin, Editor

OS/2 Developer: Vol. 5, No. 4, September/October 1993

Subscription information: U.S.: One year (six issues), \$39.95. Canada, Mexico, and international surface mail, add \$16 per year for postage. Canadian GST no. 124513185. International air mail, add \$30 per year for postage. International orders must be accompanied by payments in U.S. funds. For new orders and customer service, call (800) WANT-OS2 (800-926-8672) or (708) 647-5960 (fax: 708-647-0537). IBM employees and branch office customers can subscribe to the OS/2 Developer through IBM Mechanicsburg's Systems Library Services (SLSS) using the OS/2 Developer's order number: G362-0001. POSTMASTER: Please send address changes to OS/2 Developer, P.O. Box 1079, Skokie, IL 60076-8079. Allow eight weeks for subscriptions to begin. The OS/2 Developer is published bimonthly by Miller Freeman Inc., 600 Harrison Street, San Francisco, CA 94107, (415) 905-2200. Application to mail at second class postage rates is pending at San Francisco, CA, and additional mailing offices.

© Copyright 1993 by Miller Freeman Inc. PRINTED IN THE U.S.A.





## Spotlight

*Digitalk, a Los Angeles-based maker of software tools, has revised and improved its Smalltalk product until it has become a full 32-bit application language. The company continues to push the frontiers of objects and object-oriented programming, as **DICK CONKLIN** discovers.*

# Spotlight on Digitalk: Objects Pioneer Pushes the Envelope

**W**hile many software companies today praise the virtues of object-oriented programming (OOP) and graphical user interfaces (GUI), few can match the pedigree of Digitalk. The Los Angeles-based firm employs over 100 people in the design and development of software tools for OS/2 and other platforms. Its flagship product, Smalltalk/V, was an early offering on the original IBM PC, later became a pioneering OS/2 tool, and was recently upgraded to a full 32-bit application language.

Today Digitalk's OS/2 product line includes the OS/2 2.1-exploiting Smalltalk/V for OS/2, the Parts Assembly and Reuse Tool Set (PARTS) application development workbench, and the Team/V group development environment.

*"All original Smalltalk development takes place on OS/2."*

*—Jim Anderson, CEO, Digitalk*

### **A SPARK FROM THE PARC**

Like so many software pioneers, Digitalk was influenced by work at Xerox's Palo Alto Research Center (PARC) in the early 1970s. PARC's version of Smalltalk/V required mainframe performance and storage, but the desktop revolution of the 1980s abolished that restriction.

Jim Anderson is chairman and CEO of Digitalk and one of the company's founders. His vision for the company originated 10 years ago while he and some colleagues were working on an overseas assignment. He remembers 1983 as a time of major change in the U. S. computer industry: "We were working in Italy for Computer Sciences Corporation on a new operating system, a COBOL compiler, and tools for an Olivetti banking mini-computer. As we approached the end of the project, the IBM PC had been launched and was taking off like gangbusters. We got interested in PCs and the whole issue of personal productivity."



*Jim Anderson*

"We read the August 1981 issue of *Byte Magazine*, which was devoted to Smalltalk. At that time much of the work had taken place at Xerox PARC. To us, this language seemed ideal for developers to write PC applications; we convinced Olivetti to let us create a Smalltalk compiler for their banking system. We wrote it in Pascal."

In 1983, at the end of the project, Anderson formed Digitalk to put Smalltalk on the IBM PC. Digitalk shipped the first release in January 1985. Notes Anderson, "It was a complete Smalltalk development environment, DOS-based, floppy disk-based, called Methods. Some of the code from that product is still in Smalltalk/V today."





**Multiple Platforms, Multiple Skills.** Anderson led Digitalk from its DOS origins to OS/2, Windows, and the Macintosh. All Smalltalk development at Digitalk first takes place on OS/2; the company introduces new products on it and then migrates to other platforms. "There's even good portability," says Anderson, "from OS/2 to the Mac."

Digitalk's current product lines consist of Smalltalk/V, an object-oriented development environment, PARTS, a higher-level tool set for assembling objects and components, and Team/V, a group development and version control tool. Team/V now supports Smalltalk/V and in its August '93 release will support the PARTS product line.

Anderson claims both software vendors and corporate developers among Digitalk's customers, as well as users at various skill levels. "Smalltalk, as a language, takes more skill to use than PARTS, a higher-level tool," he explains. "PARTS lets people create applications with a lower skill level—assembling elemental components that might be created in Smalltalk, COBOL, C, or other languages."

**Object Oriented Management.** Digitalk itself is in many ways a reflection of its software philosophy. "We have small teams for each product", says Anderson. "There's a strong parallel between object-oriented programming and the change in development organizations. It's hard to build a software system that's hierarchical, with someone on the top giving orders to the rest of the program on what to do.

"Objects," he continues, "are decentralized software. Each object is an expert on its role but can utilize any other object to carry out what it wants to do.... It doesn't need to go through a chain of command to get to other objects. Our organization is just like that. There are good reasons why objects work and why human organizations work the same way; both are very efficient."



Lee Breisacher

## **SMALLTALK: THE ORIGINAL OBJECT LANGUAGE**

Digitalk's Smalltalk/V, the first commercial object-oriented language, today has over 125,000 licensed users. Its most dramatic change was made recently, not to the language itself but to its runtime environment. The 32-bit version produces applications that are up to twice as fast and half the size of their 16-bit counterparts.

**Smalltalk vs. C++.** Lee Breisacher, a senior software engineer at Digitalk, is project manager of Smalltalk/V for OS/2. He has been developing Smalltalk/V products for five years and is an ardent evangelist for the language. "I think that someone who tries C++ for a few days and then tries Smalltalk/V will choose Smalltalk/V. Go on CompuServe and read the comments from people who are using Smalltalk/V. OS/2 is a very rich operating system with hundreds and hundreds of APIs; you need to learn how they fit together and the arguments of each API. When I was using C++ I had to spend a few minutes re-learning each API each time I used it. Smalltalk/V protects you from all of that."

Mike Arrigo, Digitalk's vice president of marketing, notes, "There is a style of programming with Smalltalk/V that just isn't possible with C++. Because Smalltalk/V has an incremental compiler, you can see the results of your changes immediately; programmers are more apt to try things, to see the results of what they have done to the code." In contrast, in the C++ programming world, developers must do a re-compile and run a program before seeing where it breaks. "That alone makes Smalltalk programming a lot more productive," explains Arrigo.

Arrigo described the corporate world of the 90s as one still strongly influenced by COBOL: "C++ is still a software vendor development tool. It's much easier for COBOL-trained people to learn Smalltalk and then devote their time to learning objects. The syntax is really simple, so you can get into objects right away."



Mike Arrigo



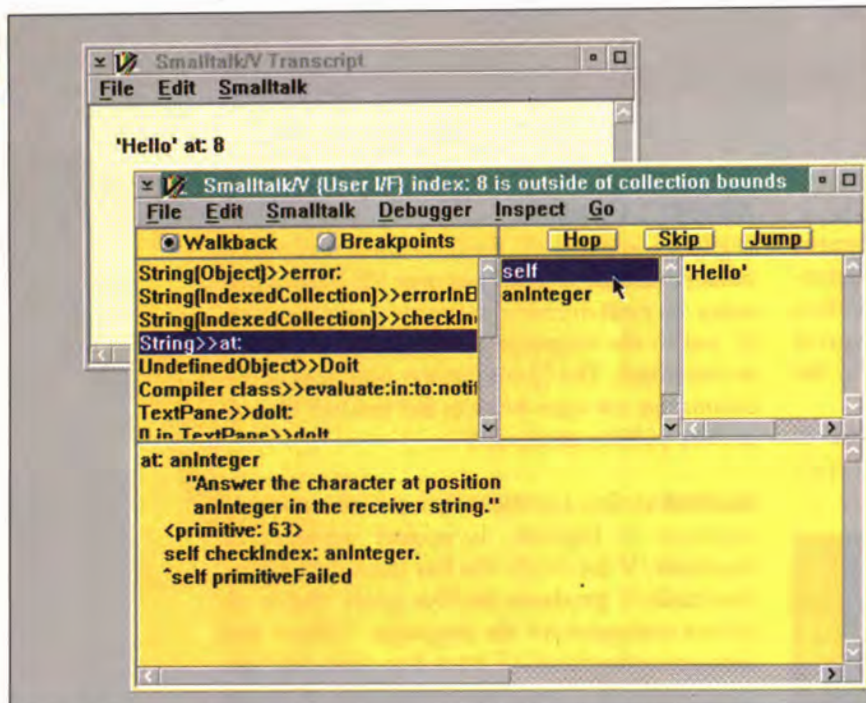


Figure 1. Smalltalk/V debugger

According to Breisacher, the original DOS Smalltalk/V was an interpretive language, and current releases still retain that "feel," even though they are compiled: "As fast as you can write a line of code you can test it. When you save your code, it is immediately compiled and stored in 32-bit 80386 binary

*"Objects are decentralized software.... Our organization is just like that."*  
—Jim Anderson

code, ready to run. You spend all your time—editing, running, debugging—in Smalltalk/V.... You can even edit your code from the debugger." (The Smalltalk/V debugger is shown in Figure 1.)

**Who Uses Smalltalk/V?** Jim Anderson describes many of Digitalk's customers as corporate shops using Smalltalk/V

to "build the client side of client/server applications." He says that customers use the language for "mission-critical applications," citing as an example one customer service system for a large utility consortium that serves two million customers."

Barbara Noparstak, director of corporate communications at Digitalk, says that the company's customers "continue to amaze us." The U.S. Postal Service has programmed its new Postal Buddy stations, to be put in 60,000 post offices, in Smalltalk/V. Explains Noparstak, "They were one of our first large customers to use our OS/2 32-bit version. We shipped our product in September '92 and in January '93 they started deploying test systems to some pilot sites."

Smalltalk/V also works as a multiplatform application tool for independent software vendors. Some .DLL files are distributed with the application as part of Smalltalk/V's runtime environment, but Digitalk charges no royalty for their use.

Says Breisacher, "We've spent a lot of time designing Smalltalk/V so it takes advantage of the environment in

which it's running. When you are running under Windows, it looks like a Windows application. On OS/2 the screens appear the way you expect OS/2 screens to look. Same for the Mac." The company's next release, he adds, will make it easier to exchange code among the three platforms.

Noparstak notes that Smalltalk/V is gaining a following among software vendors: "They find Smalltalk easier. Intersolv, for example, tried C++ but ran into too many problems. It was too complex and they weren't using objects properly. So they went to Smalltalk/V and got the job done in less time." Vendors are also attracted by the fact that they don't have to pay royalties for applications built with Smalltalk/V or PARTS.

**Exploiting OS/2 2.X.** Breisacher describes the upcoming version of Smalltalk/V for OS/2 2.x as having "true OS/2 thread support and exception handling. You don't have to use C code anymore to do those things. You can read from a named pipe and that thread will wait for something to come in. There's also a new feature called Call-In; you'll be able to write Smalltalk/V .DLLs as in



Barbara Noparstak

any other language and call them as sub-routines from a C++ program."

Breisacher also gives some hints for the future: "We're working with IBM people in Austin on SOM support, which

will ship later this year. Someday," he promises, "you'll even be able to write SOM objects in Smalltalk/V."

#### **PARTS: DIGITALK'S HIGH-LEVEL APPLICATION TOOLSET**

PARTS is getting a lot of attention—and resources—at Digitalk these days. Jim Anderson likes to describe the PARTS Workbench, the framework for



the PARTS product line, as a tool that "provides ease-of-use for programmers the way a GUI provides it for the user. Just as the notion 'If I've seen one application, I've seen them all'—same file menu, same edit menu, same drag-and-drop—works for users, we want to do the same for developers. We want 'If I've seen one part, I've seen them all'—whether [the product] is a relational database, a CICS transaction, or a SOM class. That makes application development a whole lot easier."

Anderson says that PARTS is "set above the 'language wars'", designed so that the underlying elemental parts can be built in C, C++, COBOL, or Smalltalk. "In the corporate world we find COBOL very popular; a lot of well-tested, proven code is still in use. No need to write new code—corporate

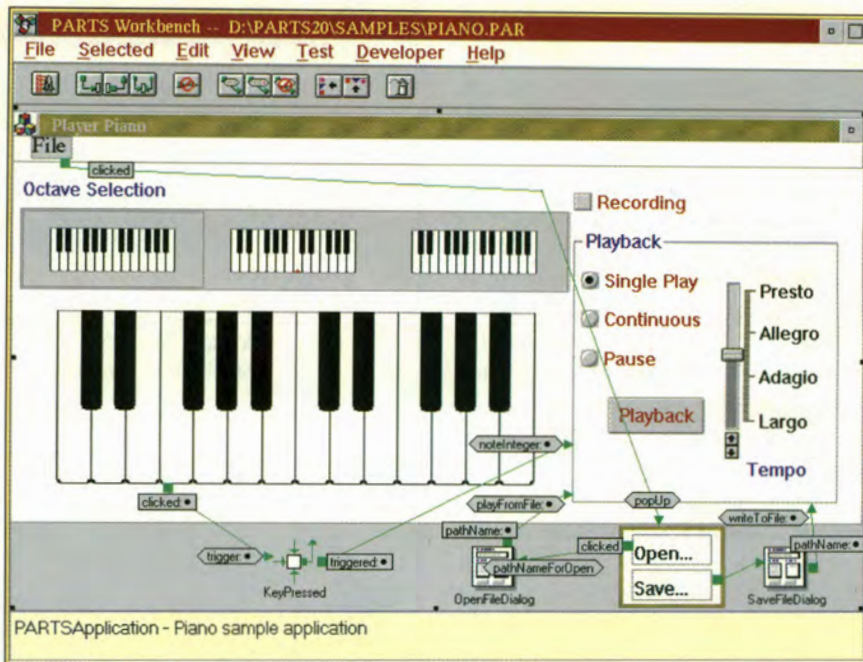


Figure 2. Parts Workbench; labels show events and messages

## Do these quotes sound familiar?

"It doesn't crash in the debugger!"

"I can't reproduce it!"

"Why does WinDefWindowProc generate an error?"

"Exactly, what did you do?"

"Where should I put WinGetLastError?"

"It must be a configuration problem!"

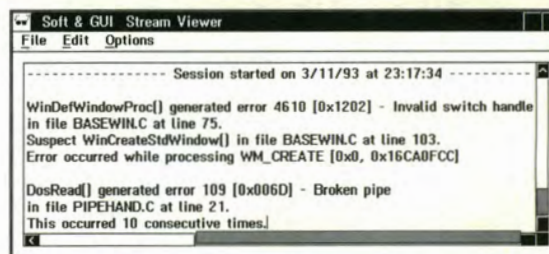
## Introducing the API Debugger which gives Answers, not Guesses

### Error Manager

32-bit  
Version 2.0  
for OS/2 2.X

Include 1 header file.  
Link 1 DLL.  
Set 1 environment variable.

- Finds intermittent errors without the Debug Kernel.
- Logs messages to a file, pipe or our PM viewer.
- Works with optimized code in realtime situations.
- Notifies your window procedure or callback function
- Supports both PM and Fullscreen programs
- Enables event-driven error handling
- Invaluable for Debug, QA Test and Production cycles



Includes a PM Viewer,  
Pointer Validity API and a  
Free copy of Command Line

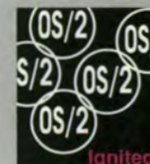
**\$225**



# Soft & GUI

2224 East 21st Street  
Brooklyn, New York 11229

(718) 769-8017





America can join the world of objects, starting with what they've got." COBOL programmers, he asserts, can make the transition to Smalltalk/V much more easily than they can to C++."

Anderson says that Digitalk's PARTS experience is proving that the 'programming' skill requirement can be lowered. Applications can be assembled from pre-fabricated parts; it is possible to construct a functioning application without writing any code, only drawing lines between events and messages, as shown in Figure 2.

**A New Software World of Parts.** Anderson uses the analogy of an automobile to explain the nesting of parts to control complexity: "You can have parts assemblies. My car engine is a part, which has several parts, such as the carburetor, which has parts of its own, and so forth. [This metaphor] is a good way to think about and solve bigger problems."

Another way to solve the big problems is with distributed objects, Anderson continues. "Today, most objects are on the client, but we can easily envision them on the server too. We'll have the flexibility of redeploying the objects. This is what distributed SOM, or DSOM, is all about; it's the underlying plumbing that makes it work. There is an opportunity for someone to develop higher-level tools, object adminis-

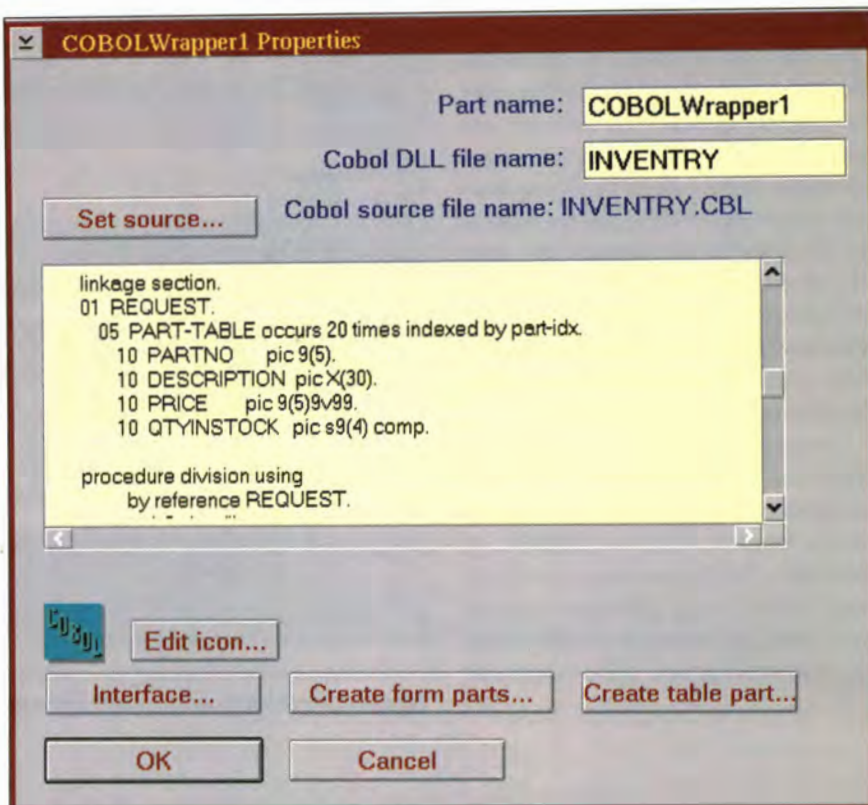


Figure 3. Property dialogue for a COBOL part

Anderson next envisions the creation of "business parts, objects with relevance to the operation of a business such as customer support, accounting, and so forth. If you understand how your business is changing, you'll be able to change and evolve your applications to reflect it." He praises the Workplace Shell's implementation of object-oriented user interface princi-

tometer object, drop it on a purchase order and it's all filled out. This will be commonplace."

Mike Arrigo calls the client/server world "a lot of transaction processing, mission-critical applications, just as on the mainframe. [Applications] may come in many flavors and not dovetail naturally. Each part, whether in SQL, COBOL, or CICS, is created and run by people in the organization who know it best. Our wrappers let developers present their data graphically on the workstation without recoding. This is more than a 'screen scraper' product, where you capture a host screen and just redisplay it graphically." (A property dialogue for a COBOL part is shown in Figure 3.)

*"We took all of the CUA '91 controls and created a PART for each one, so you can just drag and drop a notebook or slider control right onto your screen."*

*—Mike Teng*

tration tools. Now that I've got objects all over the place, where are they, what are they doing, are there any problems?"

ples and imagines how easily it could be adapted to the concept of business objects: "I can do drag-and-drop with business applications—drag a cus-

#### THE PARTS PEOPLE

We talked with several key members of Digitalk's PARTS development team: Mike Teng, vice president, Deborah Lewis, senior software engineer and team leader of PARTS Workbench for Windows, and Alex Dionysian, senior software engineer and team leader of



# GET ON A CLIENT/SERVER TRACK WITHOUT DERAILING YOUR DEVELOPMENT

If you're client/server bound, KASE:VIP visual design and code generation tools get your applications on an OS/2 or Windows GUI-based track — while leveraging your current development assets, your existing staff and even your existing application logic.

KASE:VIP automatically generates all the source code you need to build seamless links between graphical interfaces and SQL databases or CICS transactions. In standard languages like C, C++ and COBOL, so you can avoid the derailing overhead and restrictions of proprietary languages, runtimes and royalties.

KASE:VIP also offers an OPEN ARCHITECTURE license that gives you the flexibility to control the code generation process and accelerate your development. You can prototype and automatically generate code unique to your needs — such as specific line-of-business functions, proprietary APIs or your own application "hooks." All of which lets you leverage the expertise of key developers across your entire organization.

Avoid development derailment. Use KASE:VIP to keep your client/server development on the fast track.

GET YOUR GUI-BASED SQL AND CICS client/server development on a COBOL, C or C++ fast track — without proprietary languages or runtimes. Call **1-800-888-4335** for a KASE:VIP demonstration disk, or for information about KASEWORKS seminars in your area.

## KASEWORKS™

△△△△ Enabling Client/Server Strategies

(FORMERLY CASEWORKS)

3295 RIVER EXCHANGE DRIVE, SUITE 430

NORCROSS, GA 30092

**(800) 888-4335**

(404) 448-4240





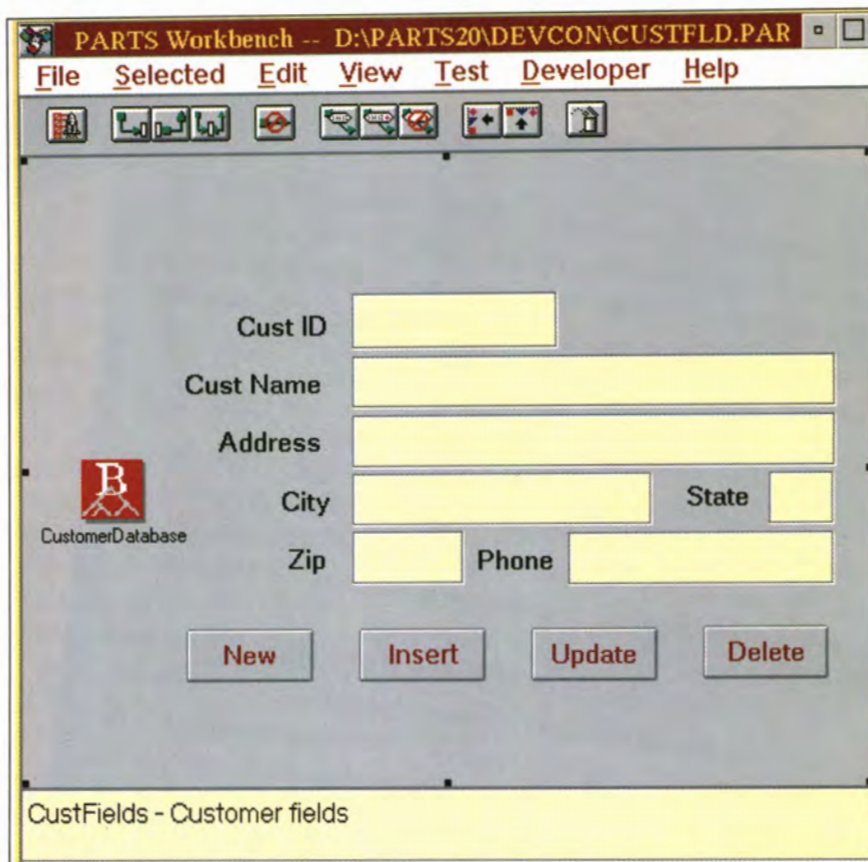
PARTS Workbench for OS/2. They explain the background and future of PARTS.

*OS/2 Developer: How did the PARTS concept come about?*

Mike: Originally, we set out to create a graphical layout tool for building GUI applications. Then we realized that layout alone wouldn't be enough; other products do that. So we created links, a visual way to link events to messages, as Smalltalk/V does.

Deborah: We also wanted this tool to be extensible. Up to now, most graphical tools haven't done very well.

Alex: One of our basic concepts is that we want to provide tools that match the complexity of the calculation—a separation of skills. Some functions, such as building the GUI, are simple and there are simple tools for them. Others are inherently complicated and require more complex tools. Let's say that you felt you needed Smalltalk or C++ to express a particular function. You would encapsulate this complex part and pass it on to someone else who was building the overall application.



**Figure 4.** A nested part containing customer fields and a non-visual Btrieve part

Mike: One stumbling block to a new object programmer is learning concepts such as inheritance, encapsulation, and

polymorphism. PARTS lifts you above those concepts.

*So this application builder person isn't a programmer?*

Mike: The separation of skills means that in big companies you start with nonprogramming analysts—people who understand the process, can lay out the screens, and can specify the functions to be performed. Then the programmers go to work writing scripts or code for the individual parts. Deborah: Then they create a parts catalog that lists all the component parts. The analyst selects and wires them together.

Alex: Later, if the components all exist and the analyst decides to change the interaction with the user, that can be done without involving the programmers.

Mike: Speaking of usability, we took all

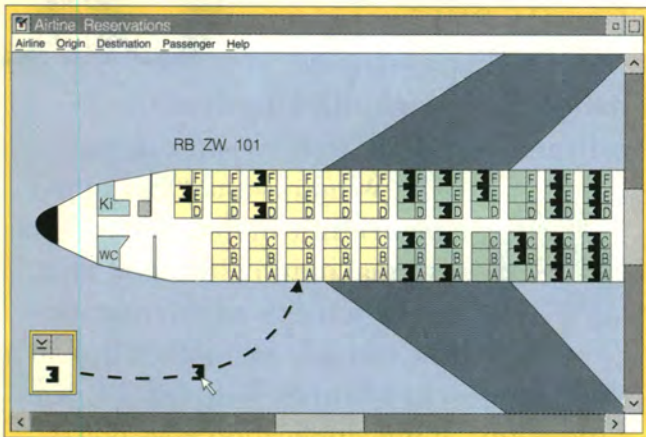


Alex Dionysian, Deborah Lewis, Mike Teng



# This program contains scenes of a graphic nature.

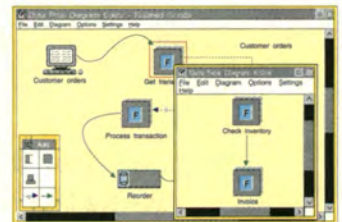
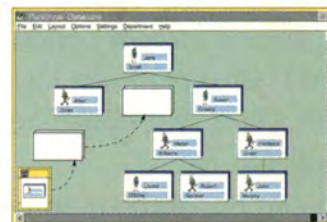
Introducing  
Graphics  
Interface  
Kit/2



But don't worry. It's all in good clean fun. Because now there's a fast, cost-effective way to create graphic, easy-to-use interfaces for OS/2® 2.0 applications and tools. Introducing Graphics Interface Kit/2 (GIK/2), an AD/Cycle® product from IBM Programming Systems.

GIK/2 makes it easy to produce and manipulate symbolic representations of data without writing a single line of Presentation Manager® code. It automatically generates C code, where you can link the graphic symbols to the application data they represent. This not only speeds up your development cycle, it also helps you create applications that are intuitive and easy-to-use. Simplify transaction systems. Make databases easy to access. Bring organizational charts to life. Track inventory with simple icons. Any application can be made easier with the right graphic front-end. And GIK/2 can help you make it happen.

For orders or additional information, please call 1 800 IBM-CALL and ask for Department S71. If you need graphic proof, ask for our evaluation demo which, by the way, is rated G.





# "No more

plus take advantage of over 250 available OS/2 development tools and utilities.

"It's a productivity thing."

"I use CA Realizer to prototype new graphical interfaces. I write code using CA C++™ and CommonView™ (integrated with IBM's C Set/2™ compiler) while I compile in the background. And when designing reports or planning schedules, I use CA-RET."

OS/2's pre-emptive multithreaded multitasking dynamically manages CPU time so you can run



DOS, Windows and OS/2 apps concurrently in different sessions with maximum efficiency. That means you can edit in one window, compile in another, link in a third and test in a fourth. With OS/2 Crash Protection™, if one application goes down due to a bug, the rest you're working on won't. "There's no limit to what you can do with this system.... It's definitely made me more productive."

## The Development Platform of Choice

- Enhanced development platform for DOS, Windows and OS/2 apps.
- OS/2 Crash Protection for superior reliability.
- Pre-emptive multitasking for increased productivity.
- Virtual memory provides up to 512MB per session.
- Flat memory model eliminates wrestling with segments.
- Multiple virtual DOS machines for concurrent app testing.
- Object-oriented Workplace Shell is easy and intuitive.



*Gary Slattery, Software Developer, Computer Associates*

"The best platform for DOS and Windows."

"I develop software applications for a living and I think OS/2® is a great way to do business." A 32-bit, virtual memory operating system, OS/2 is the ideal platform for developing your DOS, OS/2, Windows™ and even host-based applications.

With OS/2 you can boost the power of your favorite DOS and Windows tools,

IBM and OS/2 are registered trademarks and Workplace Shell, C++, CommonView, C Set/2 and OS/2 Crash Protection are trademarks of International Business Machines Corporation. All other products are trademarks or registered trademarks of their respective companies. ©1993 IBM Corp.





# arrested development."

The object-oriented user interface—the Workplace Shell™ (WPS)—gives you easy control with direct manipulation of visual objects on your computer screen. And should you need assistance with anything, IBM's Worldwide Developer Assistance Program is always there to help.

**"A developer's dream."**

With OS/2's 32-bit architecture, you can push your 386 and 486 hardware to the limit, and develop spectacular multimedia and enterprise-wide client-server applications. OS/2 lets you create the widest range of applications, for a variety of platforms, for almost any size computer. Here's your chance to develop truly revolutionary 32-bit applications. With OS/2, now there's nothing stopping you.

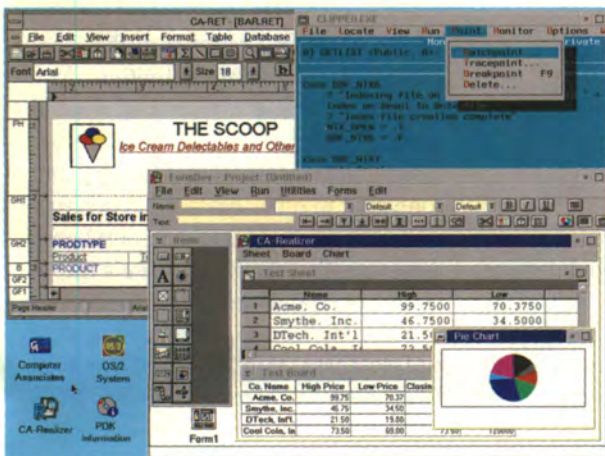
To learn more, call 1 407 982-6408 now, and get a free white paper on why OS/2 is the ideal platform for your development efforts.

**Operate at a higher level.**

1992  
- WINNER -  
PC Magazine Award  
for Technical Excellence



OPERATING SYSTEMS AND  
SOFTWARE STANDARDS  
OS/2, Version 2.0  
IBM Corporation



Use OS/2 to increase productivity of DOS, Windows and OS/2 application development.





of the CUA '91 OS/2 controls and created a part for each one, so you can just drag and drop a notebook or slider control right onto your screen. We actually wrapped, or encapsulated, the OS/2 control APIs so you are using the real thing, not a simulation.

*So some parts can contain or reference other parts?*

Deborah: The nested part facility lets the designer take a set of primitive parts and build on them. This is analogous to subroutines in conventional programming.

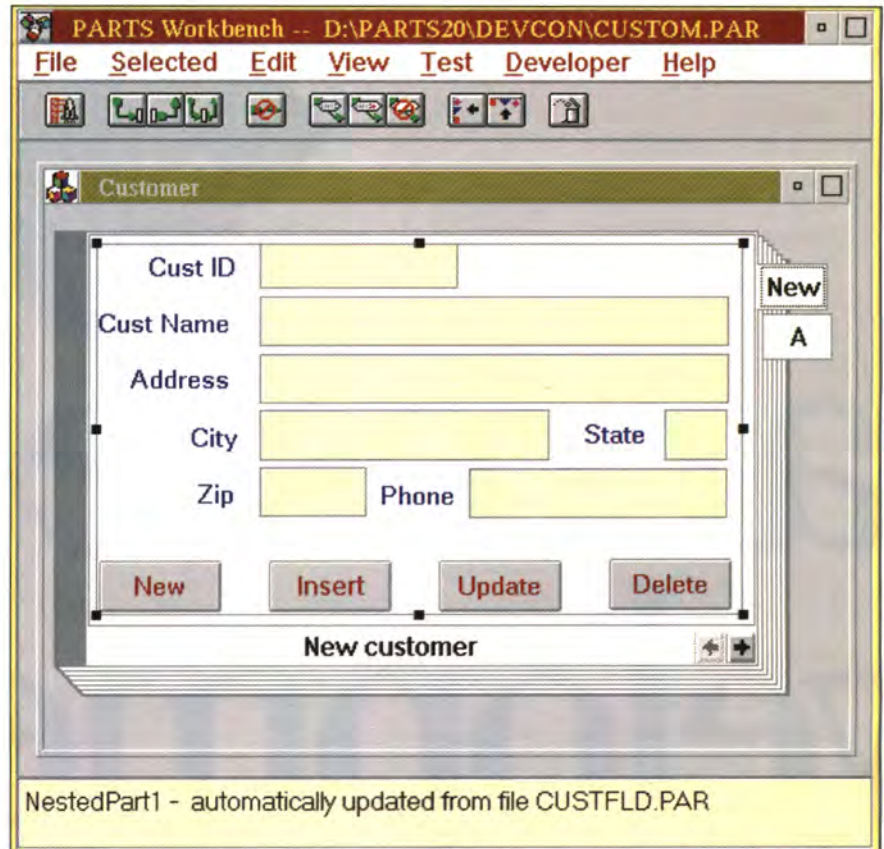
Alex: For example, you could develop a set of parts for data-entry that resemble a company's standard input forms: name, address, phone number, and so on. [Figure 4] Then you could nest those parts on pages of the company notebook. [Figure 5]

Mike: Sometimes GUI applications can result in very cluttered screens. Nested parts make it easier to pick and choose just which functions you want to appear on the screen at any point in time. Just as structured programming's nested subroutines made programs easier to read, nested parts make screens easier to use.

*"Just as structured programming's nested subroutines made programs easier to read, nested parts make screens easier to use."*

*—Mike Teng*

Alex: You still may have some application functions that are outside of your set of parts, such as a database program written in COBOL or C. These are especially important in client/server applications. We surface the functions in the external program as messages that you can link to from the PARTS Workbench



**Figure 5.** A customer application with a customer field part nested in a notebook part

and create your own interface with. We provide some of these parts as templates that are packaged and sold separately, such as our database, COBOL and CICS wrappers. Each of those

port distributed SOM objects that can be shipped from the server to the client.

### AN EXPANDING PARTS MARKET

A growing after-market includes add-on products from Digitalk and other companies. "There are several related products on the market today, and more are coming," says Anderson. "In development we have a SOM part and a CICS part. The SOM part has an underlying SOM repository that contains all the SOM class definitions. You select a class and a part is generated with the interface defined by that class. You can immediately start to use it and connect it to other parts. Each generated part is a fully functioning SOM part. For example, you could create a multimedia part and immediately open a sound or video file, play it, rewind it, and so forth. Similarly, a CICS part might be a typical transaction in a CICS application."

If you were developing a vertical



application, he adds, "you could sell your parts catalogue to other businesses in your industry. For example, a catalogue of banking components could be used as a base for other, customized, banking applications."

### TEAM/V FOR COLLABORATIVE PROGRAMMING

Team/V is an extension to Smalltalk/V that allows development teams to better organize, structure, and coordinate their work. Team/V gives Smalltalk programmers a way to structure applications into discrete pieces that can be viewed, manipulated, and maintained separately. In the future, Team/V's function will be applied to the PARTS environment. Today it is a Smalltalk/V tool.

### TEAM/V TALK

Next, we spoke to Juanita Ewing, a member of the technical staff and project leader for Team/V, based in Portland, Ore.

*OS/2 Developer: How would two or more Smalltalk/V programmers benefit from Team/V?*

Juanita: Team/V provides structural capabilities to make it easier for teams of programmers to write applications. Team/V's modular structures, called packages, are the basis of work assignments and of sharing between team members. An example package may have the function of printing, another one might do numeric calculations.

Let me make an analogy. Let's say you had a group of cartographers working on producing a map of the world. But the world isn't too stable and things keep changing. If you just turned everyone loose and let them change anything they wanted to, you would have chaos, multiple versions of the world map. The solution is to divide the map up into smaller, more manageable units. "Here, you take Europe. You take Africa." It's just a way of viewing or structuring an application.



Juanita Ewing

Furthermore, let's keep a version of each release so that you can go back in time. This is very important for traceability and accountability. "Show me the world in 1988." Team/V does this through PVCS, which has become an industry standard version control system. Source code is stored in packages (a unit of sharing) in a repository where other team members have access. This works for both vendors and corporate programming shops.

We also added PARTS access to Team/V repositories; PARTS Workbench developers can use the Team/V capabilities to save versions of parts, in addition to saving packages of Smalltalk code.

### FOR MORE INFORMATION

In addition to Digitalk's corporate and retail sales channels, IBM markets Digitalk's Smalltalk/V, PARTS, and Team/V through its Cooperative Software Program. For more information, contact Digitalk:

Digitalk Inc.  
5 Hutton Center Dr., 11th Fl.  
Santa Ana, CA 92708

Phone: (714) 513-3000  
Fax: (714) 513-3100

CompuServe: 76711,366 or  
Go DIGITALK

Internet: info@digitalk.com

**Dick Conklin**, 3408 Sherwood Blvd., Delray Beach, Fla. 33445, (407)495-4421. Conklin is the editor of *OS/2 Developer*. He recently retired from IBM after 20 years in engineering, education, marketing, and developer support. A member of the original IBM PC development team (DOS 1.1), Conklin is the author or general editor of five books on PC graphics, laptop computing, and OS/2.

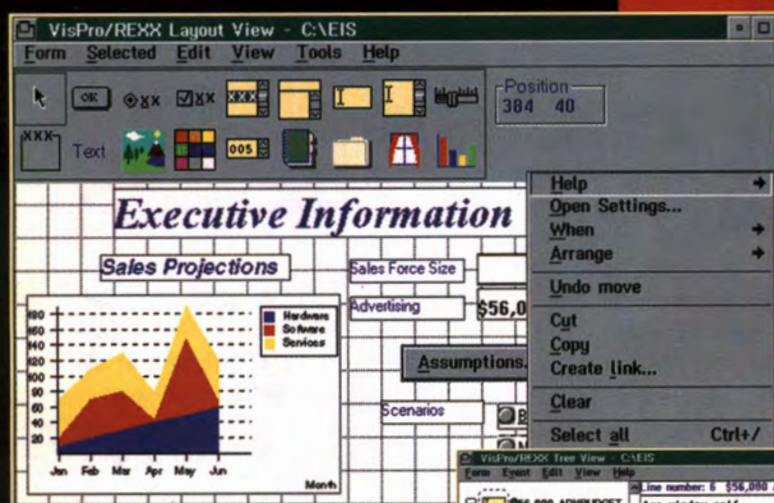




OS/2's time is here, thanks to

# VisPro/REXX™

- Visual programming
- WYSIWYG editors
- Drag & drop interaction
- Client/server programming



## VisPro/REXX

*"... brought the house down at a recent OS/2 conference in Colorado... it is to REXX and OS/2 what Visual Basic is to Windows and DOS."*

—Robert X. Cringely  
InfoWorld  
January 25, 1993

VisPro/REXX is an easy-to-use visual programming tool with a CUA '91 graphical user interface.

By using VisPro/REXX, you can develop OS/2 GUI applications in record time.

Even if you've never programmed in REXX.

And if you're an experienced programmer, you'll love the way VisPro/REXX gives you easy access to OS/2 programming features,

such as buttons, lists, sliders, graphs, charts, graphics, OS/2 Database Manager, APPC, and HLLAPI.

Free demo disk available for a limited time



## Best of all, it's small

System requirements: OS/2 2.0, 5 Mb memory, 1.5 Mb hard disk space

Sales (919) 387-7391

Tech Support (919) 380-0616

Fax (919) 380-0757

OS/2 2.0 and CUA are registered trademarks of International Business Machines Corporation.

**HockWare™**  
Formerly UCANDU Software

P.O. Box 336, Cary, NC 27512-0336  
© 1993 HockWare, Inc.

## \$299

Introductory price  
plus shipping & handling

Entry level version available for \$99  
plus shipping and handling





This article describes Network SignON Coordinator/2 (NSC/2), a productivity aid that provides a single logon and password change capability for users who deal with a variety of LAN and host accounts. **BY BILL BANNING, MICHELLE BUENKER, GARY HORN, and JIM WANGLER**

# Network SignON Coordinator/2: A Plus For Network Applications

**N**etwork SignON Coordinator/2 (NSC/2) is a productivity aid that provides a single logon and password change capability for users who deal with multiple local or gateway host sessions (via 3270 or 5250 host terminal emulation, or APPC), multiple LAN servers (IBM and Novell NetWare), or one or more OS/2 Database Manager servers.

Developers can tailor NSC/2 to their work environment to increase personal productivity. Its API, command line interface, and user exits allow its functions to be incorporated into application programs. The user interface is shown in Figure 1.

## FUNCTION

NSC/2 1.1 allows users to log on to, log off from, or change passwords on a variety of systems, including hosts (VM, MVS and OS/400), local OS/2 UPM, OS/2 LAN Server domains, and Novell NetWare file servers.

**Supported Configurations.** NSC/2 can be configured as either client (OS/2 or DOS) or server. Users at OS/2 clients can benefit from the full range of functions available, while DOS users are more limited in what they can do. Figure 2 gives an overview of the product's functional capabilities.

The NSC client configuration is installed on OS/2 or DOS systems on which users initiate password update or sign-on activity. Any workstation with installed NetBIOS support can also be configured as a client and perform various remote

password operations via a remote workstation configured as an NSC server.

Systems configured as NSC servers can receive and process password change and password verification requests from DOS and OS/2 clients, providing more flexibility with regard to host connectivity and extending the capabilities of client workstations.

*NSC/2 controls password changes and access to mainframes and servers.*

The server component should be installed only on machines that will be used for one or more of the following:

- They will locally process requests for UPM password changes from other client workstations (as with OS/2 Database Manager servers). NSC, however, need not be installed on a LAN server that handles only LAN-related logon and password change requests from clients.
- They will process requests from other client workstations for password changes on hosts attached to the remote system (in which case the system acts as a gateway to a host for other DOS and OS/2 clients).
- The machine is one on which user or group



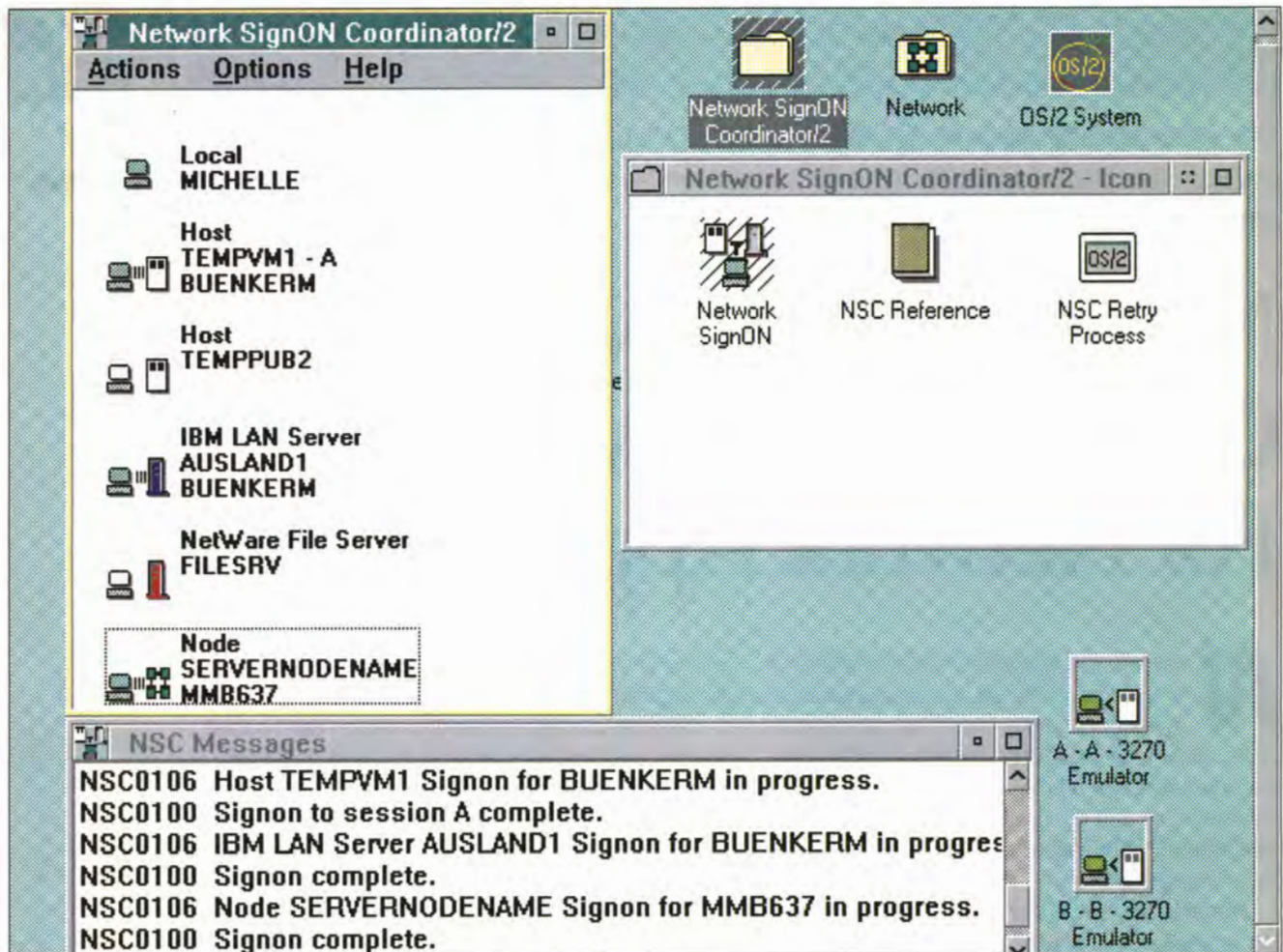


Figure 1. Network SignON Coordinator/2 1.1 user interface on OS/2

NSC program configuration information will be stored and maintained, utilizing the server's function as a central repository for those files.

nitions are available: LOCAL, LANSERVER, HOST, NODE, and SERVER.

**Scenario.** An OS/2 2.0 user has user accounts on a variety of systems—the

accessed), and an OS/2 Extended Services Database Server. The user would create one definition for each of these accounts in the NSC.INI file. A sample NSC.INI file is shown in Figure 3.

Notice that definitions that follow a SERVER statement are actually executed at the machine configured as an NSC server. In the example code in Figure 3, the database server has been configured as an NSC server and also acts as a gateway to the host (for instance, via the DDCCS/2 product). The user synchronizes the password for the OS/2 database account and the host account with the two definitions following the SERVER statement. The LOCAL statement ensures that all password changes will be done on the user account at the database server, and the HOST statement does the same for the host account.

*An OS/2 application can be launched from an NSC/2 signon screen.*

### CONFIGURING NETWORK SIGNON COORDINATOR FOR YOUR ENVIRONMENT

Signon and password change activity performed by NSC is driven by definitions in the NSC.INI configuration file that tell the program which accounts to act upon. Five major defi-

local UPM, three OS/2 LAN server domains (the user usually logs on to one but also uses resources on the other two), one NetWare file server, two local IBM host sessions (one via 3270 terminal emulation and the other via 5250), one host session through a remote gateway via APPC (through which DB2 is





Some additional configuration activity must be done for host support. On OS/2, NSC uses OS/2 Communications Manager's Emulator High Level Language Application Program Interface (EHLLAPI) to communicate with hosts connected via 3270 or 5250 terminal emulation. To tailor the logon and password change activity to their unique hosts, users create host information files using a script language provided by the program. The script language provides the flexibility for NSC to adapt to virtually any IBM host environment. Communication with hosts connected via APPC is more complex and involves creating or modifying network definitions at both the workstation and the host computer.

### **CUSTOMIZING AND DEVELOPING APPLICATIONS WITH NSC**

Applications can be integrated with NSC in two ways. First, the Presentation Manager (PM) interface can be used as the front end for an application, launching the application from user exit routines during signon, signoff, or password change activity. Keeping the PM interface active allows users to selectively sign on to and off from individual, configured locations.

An application can also provide its own user interface to obtain the user ID and password, and then invoke NSC to:

- Sign on to the user's default locations
- Sign off from all locations at which a user is signed on
- Change the password on the user's default locations.

NSC can be invoked by an application through either an API or the command line interface.

**User Exit Routines.** Its user exit capability enhances NSC's productivity in a variety of environments. Exit routines, which are automatically started when a signon, signoff, or password change request is processed in NSC.INI, can be either command files (started using the

| Network Signon Coordinator/2 Client Capabilities                         | DOS Client | OS/2 Client |
|--------------------------------------------------------------------------|------------|-------------|
| <b>Signon/Signoff</b>                                                    |            |             |
| UPM local or node logon                                                  |            | •           |
| IBM LAN Server domain                                                    | •          | •           |
| Novell NetWare file server                                               | •          | •           |
| Host attached via local emulator                                         | •          | •           |
| Host attached direct via APPC (password verification only)               |            | •           |
| Host attached through NSC Server using APPC (password verification only) | •          | •           |
| <b>Password Changes</b>                                                  |            |             |
| Local UPM                                                                |            | •           |
| IBM LAN Server domain                                                    | •          | •           |
| Novell NetWare file server                                               | •          | •           |
| Host attached via local emulator                                         | •          | •           |
| Host attached direct via APPC                                            |            | •           |
| Host attached through NSC Server                                         | •          | •           |

**Figure 2.** Functional summary

|                                                          |                                                                     |
|----------------------------------------------------------|---------------------------------------------------------------------|
| USERID=JFB                                               | Default user ID (can be overridden)                                 |
| LOCAL,ON                                                 | Sign on to local UPM account                                        |
| HOST,EMULATOR,HIF=VM1HIF.HIF,NAME=VM1,ON                 | Signon to host VM account                                           |
| HOST,EMULATOR,HIF=OS4HIF.HIF,NAME=OS400,USERID=JEFF,ON   | Signon to OS/400 using a different user ID                          |
| LANSERVER,IBM,NAME=DOMAIN1,ON                            | Sign on to DOMAIN1                                                  |
| LANSERVER,IBM,NAME=DOMAIN2<br>LANSERVER,IBM,NAME=DOMAIN3 | Keep password updated on DOMAIN2 and DOMAIN 3 (but don't sign on)   |
| LANSERVER,NOVELL,NAME=FSERVE01,ON                        | Connect to NW File Server                                           |
| SERVER,NAME=DBSERV<br>*                                  | Statements after "SERVER" will be performed on the specified server |
| LOCAL<br>*                                               | Keep password updated on database server                            |
| HOST,APPC,NAME=MVS1                                      | Keep password updated on DB2 host                                   |

**Figure 3.** Sample NSC.INI file





command interpreter as windowed sessions) or executable programs. All exit routines are started in a separate session in either the foreground or the background and may be PM, full-screen or windowed applications (in DOS, exit routines must be noninteractive).

Exit routines can be used for a variety of tasks, including assigning or cleaning up redirected drives, logging on to or keeping passwords synchronized with other applications, or executing tasks associated with signon activity.

*Exit routines can be used for a variety of tasks, including assigning or cleaning up redirected drives, logging on to or keeping passwords synchronized with other applications, or executing tasks associated with signon activity.*

Figure 4 shows a sample user exit routine that could be run on a DOS Database Client (from IBM Extended Services 1.0). Since NSC does not specifically support DOS Database Client logons, a user exit can be implemented to perform this function. Note that the user exit statement is associated with a NODE definition in the NSC.INI file.

**API Support.** NSC/2 is shipped with a small toolkit that lets programmers incorporate the program's functions

```
DOSDBLOG.BAT
-----
@echo off
rem %1 = 0 for signon, 1 for signoff, 2 for change password,
rem %2 = Configuration definition index
rem %3 = Return code
rem %4 = User ID
rem %5 = Current Password
rem %6 = New Password
rem
rem if a signon request
if not %1 == 0 goto off
rem
rem perform the database logon
sqllogn2 %4 /p:%5
goto end
:off
rem
rem if a signoff request
if not %1 == 1 goto pw
rem
rem perform the database logoff
sqllogf2
goto end
:pw
rem
rem if a change password request
if not %1 == 2 goto end
rem
rem perform the database logon using the new password
sqllogn2 %4 /p:%6
:end

NSC.INI
-----
...
NODE,NAME=IGNORED,ON,EXIT=DOSDBLOG.BAT
...
```

Figure 4. User exit for logging on at a DOS database client

```
#include "nscrsign.h"

unsigned char UserId[NSC_MAXUIDLEN+1]; /* User ID */
unsigned char Password[NSC_MAXPWLEN+1]; /* Current Password */
unsigned char NewPassword[NSC_MAXPWLEN+1]; /* New Password */
short Action; /* NSC_LOGON, NSC_LOGOFF, /*
/* or NSC_CHGPW */
FNMSGSP *msgdsp; /* Message Callback Proc */
short return_code; /* Return Code */

return_code =
    NSCRSIGN (UserID, Password, NewPassword, Action, NULL, msgdsp);
```

Figure 5. NSCRSIGN API syntax



# Mission



IBM Programming Systems introduces C Set++,™ the most complete application development package you can buy for OS/2®. Its 32-bit C/C++ compiler lets you unleash all the power of OS/2 — so you can create the most advanced, high-performance applications.

It has an extraordinary code optimizer with a full set of options. Even a switch to optimize for the new Pentium™ processor. Plus a full set of class libraries, including application frameworks for PM, container classes and classes for multitasking, streams and more.

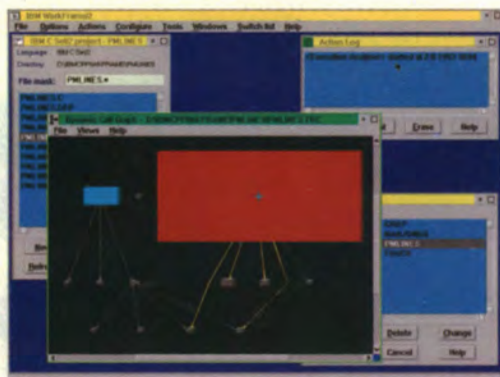
There's also a full complement of other helpful features. Such as an interactive source level debugger.



## C Set ++

shows class library relationships.

What's more, you get Workframe/2,™ a language-independent tool that lets you customize your own environment. It's adaptable and flexible — you can use any 16 and 32-bit DOS, Windows™ and OS/2 tools.



To order C Set++,  
**contact your nearest dealer or call**  
**1-800-342-6672 (USA) or**  
**1-800-465-7999 ext. 460 (Canada).**

Clearly, there's only one place to start. C Set++.

# critical code

### C Set ++ Technical Features

|              |                           |
|--------------|---------------------------|
| Standards    | ANSI C X3.159-1989        |
|              | NIST validated            |
|              | ANSI C++ X3J16 (Full ARM) |
|              | ISO 9899:1990             |
| Optimization | Global                    |
|              | Inter-module              |
|              | Function inlining         |
|              | Instruction scheduling    |

# starts here.



# MAKE YOUR OS/2 POWER MOVE NOW!

*for applications that are*

• more dynamic

• easier to use

• faster to develop



NEW AND BESTSELLING RESOURCES from



**VAN NOSTRAND REINHOLD**

**New!**

## THE OS/2® 2.1 CORPORATE PROGRAMMER'S HANDBOOK

By Nora Scholin, Martin Sullivan, and Robin Scragg  
Simplify migration from DOS and Windows to OS/2 2.0 and find valuable solutions based on modular subdesigns, from function key assignments to pull-down menus.

**\$39.95, 352 pages, cloth, ISBN 0-442-01598-4**

**New!**

## USING WORKPLACE OS/2®

Power User's Guide to IBM's New Operating System / 2 Version 2.1

By Lori T. Brown and Jeff Howard

Get the "inside" help of the Workplace Shell's lead designers to convert easily and quickly from Windows and Mac environments to OS/2 • set up, maintain, and customize your own Workplace environment • power up with OS/2's Multimedia Presentation Manager/2 under Workplace.

**\$24.95, 400 pages, paper with disk, ISBN 0-442-01590-9**

## WRITING OS/2® 2.0 DEVICE DRIVERS IN C

By Steven J. Mastrianni

The first guide to programming 32-bit OS/2 device drivers includes C source code examples, with optional disk. Reduces the difficulty and cost of writing device drivers for OS/2.

**\$36.95, 410 pages, paper with optional disk, ISBN 0-442-01141-5**

**New edition of a best seller!**

## CLIENT/SERVER PROGRAMMING WITH OS/2® 2.0

Second Edition

By Robert Orfali and Daniel Harkey

Readers raved about the first edition of this definitive Client/Server-OS/2 resource. In-depth tutorials and sample code make this the ideal guide to client/server in the 32-bit environment. Covers all new 2.0 functions.

**\$39.95, 1,026 pages, paper, 0-442-01219-5**

**New!**

## OS/2® 2.X NOTEBOOK

The Best of OS/2® Developer Magazine

Edited by Dick Conklin

Foreword by Philippe Kahn, Borland International, Inc.

Here is a wealth of practical tips and techniques, coding examples, and product reviews—all designed to help you harness the power of OS/2. An OS/2 resource worth its weight in gold!

**\$34.95, 1,164 pages, paper, 200 illustrations, ISBN 0-442-01522-4**

**New!**

## THE SHELL COLLECTION

OS/2® 2.X Utilities

By Steve Levenson

This disk and text provide a range of OS/2 2.X software that may not be available in local computer stores or from mail order companies. "Try before you buy" utilities such as • Golden Compass • OPTICACH • SEESYS • SETSWAP • Open shutter • Window Washer • 40S2 • LIGHT • DDUMP

**\$29.95 paper, 128 pages, ISBN 0-442-01585-2**

**EASY ORDERING! CALL TOLL-FREE:**

**1-800-544-0550 (and get a full listing of VNR titles).  
OR FAX 1-606-525-7778**

**NEW! Coming in September**

Visit the **VNR COMPUSERVE BOOKSTORE**  
for a more extensive listing of VNR titles.  
*Watch for details!*



**VAN NOSTRAND REINHOLD**

*Publishing for Professionals Since 1848.*





#### Procedure Definition:

```
void NSC_DLL_ENTRY
    dspmsg (short          config_index,
            short          return_code,
            short          reserved,
            unsigned char * NSCPTR message_text)
```

#### Parameter Definition:

```
config_index -- Index of the LOCAL, LANSERVER, HOST SERVER or NODE
               definition being processed
return_code  -- Error return codes defined in toolkit header
reserved     -- Not used
message_text -- Message text associated with error
```

Figure 6. Message callback procedure

```
#include <stdio.h>
#include <stdlib.h>
#include "nscrsign.h"

/* Message callback procedure */

void NSC_DLL_ENTRY
    dspmsg (short          config_index,
            short          return_code,
            short          reserved,
            unsigned char * NSCPTR message_text);

int main (void);

/* Enable 32-bit definitions */

#ifdef NSC32
#pragma linkage (dspmsg, far16 pascal)
#endif

int main ()
{
    short rc;

    /* Call the NSCRSIGN API to log the userid USERID with password */
    /* PASSWORD on to all systems defined in NSC.INI */

    rc = NSCRSIGN ("userid", "password", NULL, NSC_LOGON, NULL, dspmsg);
    printf ("NSCRSIGN rc = %d\n", rc); /* Print the return code */
}
```

Figure 7. Application's use of NSCRSIGN API (continued on page 26)

into their own applications.

The NSCRSIGN API allows 16- and 32-bit applications to take advantage of NSC's capabilities. The application passes the API the type of request (signon, password change, or signoff), user ID, password, and the address of a procedure to be called for messages generated by the API. The application can display these resulting messages, write them to a log file, or perform other processes based on return codes associated with each message. Figure 5 depicts the syntax of the NSCRSIGN API.

Errors from the NSCRSIGN API are reported to the calling application via a message callback procedure. The address of this procedure is passed as a parameter to NSCRSIGN, thereby avoiding interaction with the user. This allows the use of either PM or full screen applications.

The message callback procedure should be defined as shown in Figure 6. Figure 7 illustrates a call to NSCRSIGN with all messages written to standard output. (If using the IBM C Set/2 compiler, use the compiler option -DNSC32 to enable 32-bit definitions in the toolkit header file as well as in your source code.)

**Command-Line Interface.** NSC's command line interface allows a REXX command file to take advantage of the program's capabilities. The REXX command file calls the command-line interface with the type of request (signon, signoff, or password change), user ID, and optionally a password and new password. NSC returns a return code indicating the overall success or failure of the call. Messages are written to standard output and may be redirected to a file for more sophisticated error analysis.





**William L. Banning**, IBM Personal Software Products, 11400 Burnet Rd., Austin, Texas 78758, (512) 838-0185. Banning is a senior programmer with IBM PSP LAN Systems. He holds an M.S. in computer science and a B.S. in mathematical statistics from the University of Missouri.

**Michelle M. Buenker**, IBM Personal Software Products, 11400 Burnet Rd., Austin, Texas 78758, (512) 838-0405. Buenker is a staff programmer for IBM PSP LAN Systems. She joined IBM in 1987 and holds a B.A. in computer science from Baylor University.

**Gary R. Horn**, IBM Personal Software Products, 11400 Burnet Rd., Austin, Texas 78758, (512) 838-0271. Horn is a senior programmer for IBM PSP LAN Systems. He joined IBM in 1977. From 1986 to 1992, he worked on OS/2 Database Manager, serving for four years as the chief programmer for database management products development. Horn received a B.A. in mathematics and a M.A. in computer science, both from the University of Texas at Austin.

**James A. Wangler**, IBM Personal Software Products, 11400 Burnet Rd., Austin, Texas 78758, (512) 838-0091. Wangler is a senior associate programmer for IBM PSP LAN Systems. He holds a B.S. in computer science from North Dakota State University.

```
return (0);
}

/* Message callback procedure definition */
/* This procedure simply lists the 3 informational details returned */
/* by the API: the index of the NSC.INI definition associated with */
/* the error; the error return code; and the actual message text. */
void NSC_DLL_ENTRY
    dspmsg (short          config_index,
            short          return_code,
            short          reserved,
            unsigned char * NSCPTR message_text)
{
    (void) reserved;
    if (message_text & 1brk.0 & 1brk.)
    {
        printf ("%3d %3d %s\n",
                config_index, return_code, (void *) message_text);
    }
}
```

Figure 7. Application's use of NSCRSIGN API (continued from page 25)

## REFERENCES

For a full description of this program, see the *IBM Network SignON Coordinator/2 Online User's Guide* included in the program package. Functional highlights and software prerequisite information can be obtained from the IBM Fax Information Service, (800) 426-4329 in the U.S. or (415) 855-4329 outside the U.S. Requests for information and orders can also be submitted by calling (800) 426-2255 in the U.S.

A starter booklet, *IBM Network SignON Coordinator/2 Version 1.1* (IBM Doc. S96F-8629-00) is also available.





## TLIB<sup>TM</sup> Version Control For DOS, OS/2 and Windows-NT

### • The experts loved TLIB 4:

"...amazingly fast... TLIB is a great system." **PC Tech Journal**  
"TLIB has features and power to spare... TLIB is easy to use and the fastest of the reviewed packages." **Computer Language**  
"I will not program without it." **Uptime Magazine**

### • Now TLIB 5.0 adds:

Automatic branching. Automatic version labeling across branches. User defined **promote** structures, for staged development. Exclusive whole-level **change migration** for customized software. N-way-tree version numbers. Branch and full locking. OS/2 & NT support.

### • Plus the features they loved in TLIB 4:

Check-in/out locking. Branching, for parallel development. Keywords. Full binary file support (does not depend upon CRs in the file like other products). Wildcard and list-of-file support; can create lists by scanning source code for includes. Can merge (reconcile) multiple simultaneous changes and undo intermediate revisions. Network and WORM optical disk support. Mainframe-compatible delta generator for Pansophic, ADR, IBM, Sperry formats. Includes integrated PD MAKE by L. Dyer; also integrates with Opus<sup>TM</sup> MAKE, Slick<sup>TM</sup> MAKE, others.

MS-DOS \$139, OS/2 & NT (with MS-DOS) \$195 + shipping.  
5 station network: MS-DOS \$419, OS/2 \$595. Call for other sizes.



Burton Systems Software

PO Box 4156, Cary, NC 27519 (919) 233-8128  
FAX: 233-0716

Includes  
32-Bit  
OS/2 2.x  
DLL

\$159

## dbfLIB

Programmer's Library

### Accessing dBASE files has never been easier!

#### Features:

-  Royalty free DLLs for OS/2 and Windows
-  File and record locking APIs
-  Handle based calls
-  Memo field & index support
-  Static libraries for DOS, Windows, and OS/2

dbfLIB gives C programmers the easy task of accessing dBASE files while maintaining the speed and size of C programs. The APIs in dbfLIB are fully compatible across DOS, Windows, and OS/2.

Visa and MasterCard accepted.



dSOFT Development Inc.  
4710 Innsbruck Drive  
Houston, TX 77066  
(713) 537-0318

# SQL Objects ++<sup>TM</sup>

Database Class Library

## Inherit The Power<sup>TM</sup>

**SQL Objects++ Database Class Library** is a powerful collection of object-oriented database classes that provide *full* access to SQL and non-SQL databases:

- Oracle
- Sybase
- SQL Server
- Netware SQL
- Btrieve
- DDCS/2 (DB2, SQL/400, SQL/DA)
- Others

## The Object-Oriented

**Approach.** No longer are you restricted to the lowest common denominator factor when developing database applications. Inherit the power *now*. Derive new classes with powerful custom extensions. Realize more flexibility than ever before. Use Object Technology today.

## Multi-Platform Support.

Whether you use OS/2, DOS, Windows or NT, SQL Objects++ Database Class Library is designed to take advantage of your native environment.

## NO RISK and NO ROYALTIES.

Use it for 30 days. If you not satisfied, return it for a full refund. Order today for only \$499.

SPECIAL

\$249

Limited Time Offer

**GSI** Graphical  
Software  
Interfaces, Inc.

47 Stonewall Street Cartersville, GA 30120

Sales: (800) 876-6585

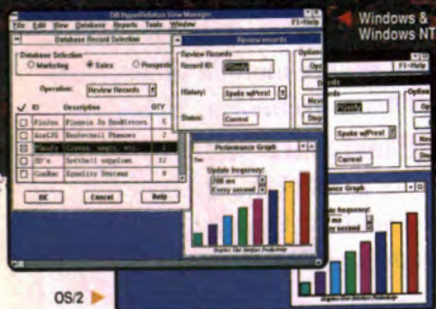
(404) 382-6585 fax (404) 382-6374

SQL Objects++ and Inherit The Power are trademarks of Graphical Software Interfaces, Inc. All company and product names are trademarks or registered trademarks of their respected owners.

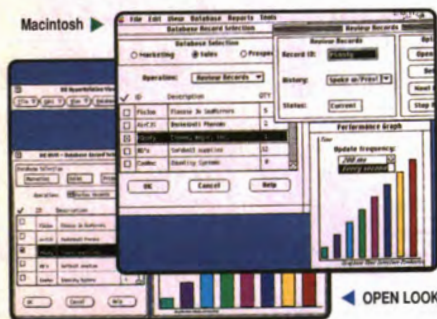


WINDOWS  
WINDOWS NT  
OS/2  
MACINTOSH  
OPEN LOOK  
OSF/MOTIF  
CHARACTER SYSTEMS

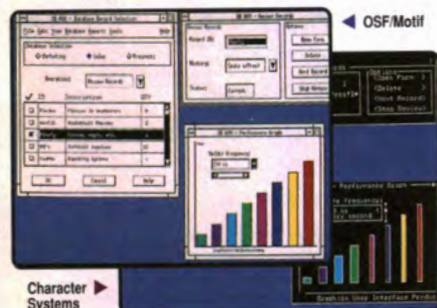
# Develop Cross-Platform GUI Applications in a Single Stroke.



OS/2



Macintosh



Character Systems

## Master the art of multi-platform GUIs.

XVT Software is the leading choice of world-class developers for one reason: It is the simplest, quickest path to building quality applications that port to every GUI without compromises in look-and-feel or performance. Plus, it's easier to learn and use than native toolkits, so your time and effort goes into your application, not your GUIs.

## XVT gives you simultaneous original GUIs.

Because XVT uses native GUI objects, your application is indistinguishable from one written directly to the native toolkit. Through our layered architecture, you achieve equivalent cross-platform functionality appropriate to each GUI, without the overhead and inflexibility of proprietary emulation-based systems.

## XVT puts complete C/C++ solutions at your fingertips.

The XVT Solutions for C and C++ each include an Interactive Design Tool and the XVT Portability Toolkit.

Our Design tools let you use your mouse to design and lay out your GUI, using native and custom controls, then test it on all your target platforms before generating and compiling your code.

When combined with in-depth consulting, training and support, plus a wide range of Partners products, XVT

forms the most comprehensive and advanced solution for developing completely portable GUI applications.

## Developers judge XVT to be a masterpiece.

XVT is the base document for the IEEE's GUI standardization effort. Our thousands of customers include internal and commercial developers like: Alcoa, Amoco, AT&T, Avis, Ford, General Motors, Grammatik/Reference Software, Kodak, Lockheed, NCR, NEC, NIST, Novell, Rockwell, Siemens, Sony, Southwestern Bell, Tandem, Uniplex, Unisys, US Army and US West.

## Call now for a free XVT Technical Overview and Demo.

Ask about XVT training in FL, NY, WA, TX, NJ, and more.



The portable GUI development solution.

**1-800-678-7988**

XVT Software Inc. 4900 Pearl East Cir. Boulder, CO 80301

(303) 443-4223 FAX (303) 443-0969

For European inquiries, contact: Precision Software GmbH

Phone: 49 0 61 03/37 94 0 Fax: 49 0 61 03/36 95 5





This article describes current remote LAN access technology along with specific features of the IBM LAN Distance product, introduced this fall by IBM. **BY PAT SCHERER**

# The LAN Distance Product: LAN Application Transparency For the Remote User

**R**emote LAN access is a rapidly growing market already beginning to spawn new applications and influence application design. It also greatly influences how, and perhaps more important, where many of us do our work by breaking the tethers of the *local* area network.

IBM's LAN Distance product is a new family of communication products that let remote users transparently run LAN-based applications over switched connections (asynchronous, synchronous, and ISDN) using public switch telephone networks or PBX/CBX exchanges. The primary distinction of the LAN Distance product is that it uses device driver replacement technology to provide a superset of functions using non-dedicated, non-proprietary hardware.

The LAN Distance product extends the LAN environment, with all of its applications and resources, to any user with access to a telephone. Most widely-used protocols and network operating systems are supported. In addition, the product is designed to be readily extendable to new applications, hardware, and protocols.

A remote workstation can connect to other remote or LAN-attached workstations. The LAN workstations are accessed via a software-based bridging, routing, and administrative component called the Connection Manager. The LAN Distance product's connection manager supports up to 32 simultaneous communication ports and provides a wide range of configurable security and administrative features.

## REMOTE LAN ACCESS TECHNOLOGIES

Other available remote LAN access products vary widely in cost and functions. The degree of functionality is decided in part by the technology chosen to provide remote access. The LAN Distance product uses a device driver replacement approach called "remote node." Two other common approaches are referred to as "remote control" and "remote client."

With the remote control approach, a remote workstation dials into and takes control of a LAN-attached workstation. The LAN-attached station executes programs on behalf of the remote station. Only keyboard and screen data from the dedicated LAN-attached system is routed back to the remote workstation, minimizing the traffic flowing across the link. This approach has some disadvantages, however. A dedicated LAN-attached workstation is required for each remote workstation dialing onto the LAN, application transparency is not always preserved, and graphical interfaces, when supported, are not supported efficiently.

The remote client approach is generally an extension of a network operating system that allows a remote client workstation to share data and applications located on a common network server. Application transparency and graphical interface support are generally preserved within the remote workstation. Disadvantages stem from the fact that the remote workstation is addressable only by the server to which it is connected; accessing another server generally requires the user to



Pat Scherer



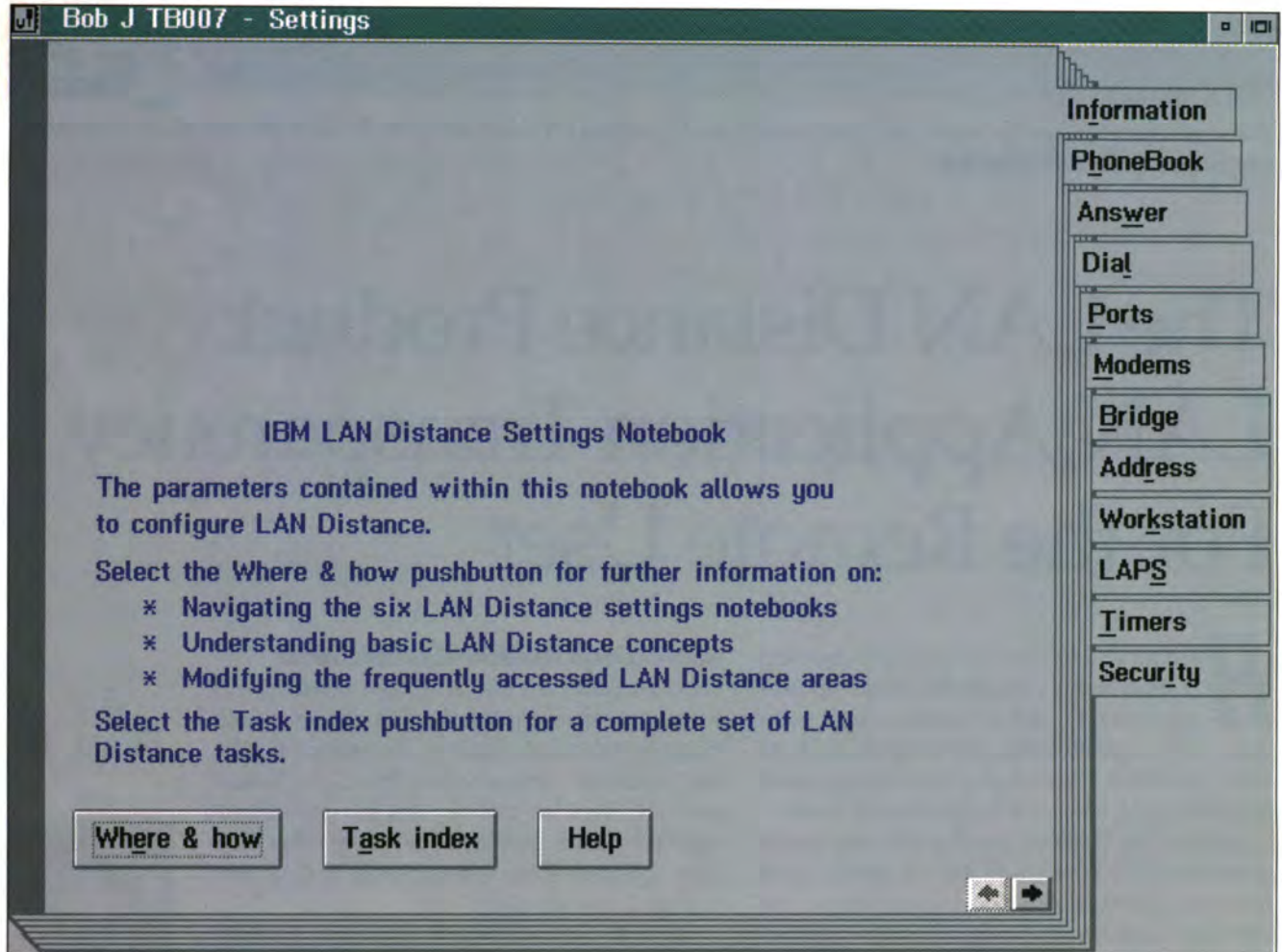


Figure 1. Settings notebook

disconnect and redial. This approach does not scale well to support of large or distributed environments, as bottlenecks in memory and CPU capacity tend to form in the common server. Because of this, most products using

node approach provides a more flexible, scalable solution for transparent remote LAN access. This approach replaces the device driver within a LAN-attached workstation (the connection server) and allows it to move data between the LAN

information and services wherever they reside on the LAN, without the necessity of redesigning the LAN with a central dedicated server to accommodate remote access. Increasing numbers of local and remote LAN users can be accommodated without duplicating (and maintaining) data files across numerous communication servers.

## *Remote LAN access influences how and where we do our work by breaking the tethers of the local area network.*

the remote client approach are dedicated servers supporting a limited number of remote connections (usually fewer than 16).

The LAN Distance product's remote

and WAN. Remote node also provides full addressability, which allows a remote workstation to access distributed LAN-attached servers and peer services. The remote workstation can access

### **ABOUT THE PRODUCT**

The LAN Distance product consists of two packages, IBM LAN Distance Remote and IBM LAN Distance Connection Server. The LAN Distance Remote package contains the code necessary to dial and accept calls from other remote workstations. The Remote package can provide a low-cost means for LAN applications to communicate without a physical LAN: if installed on



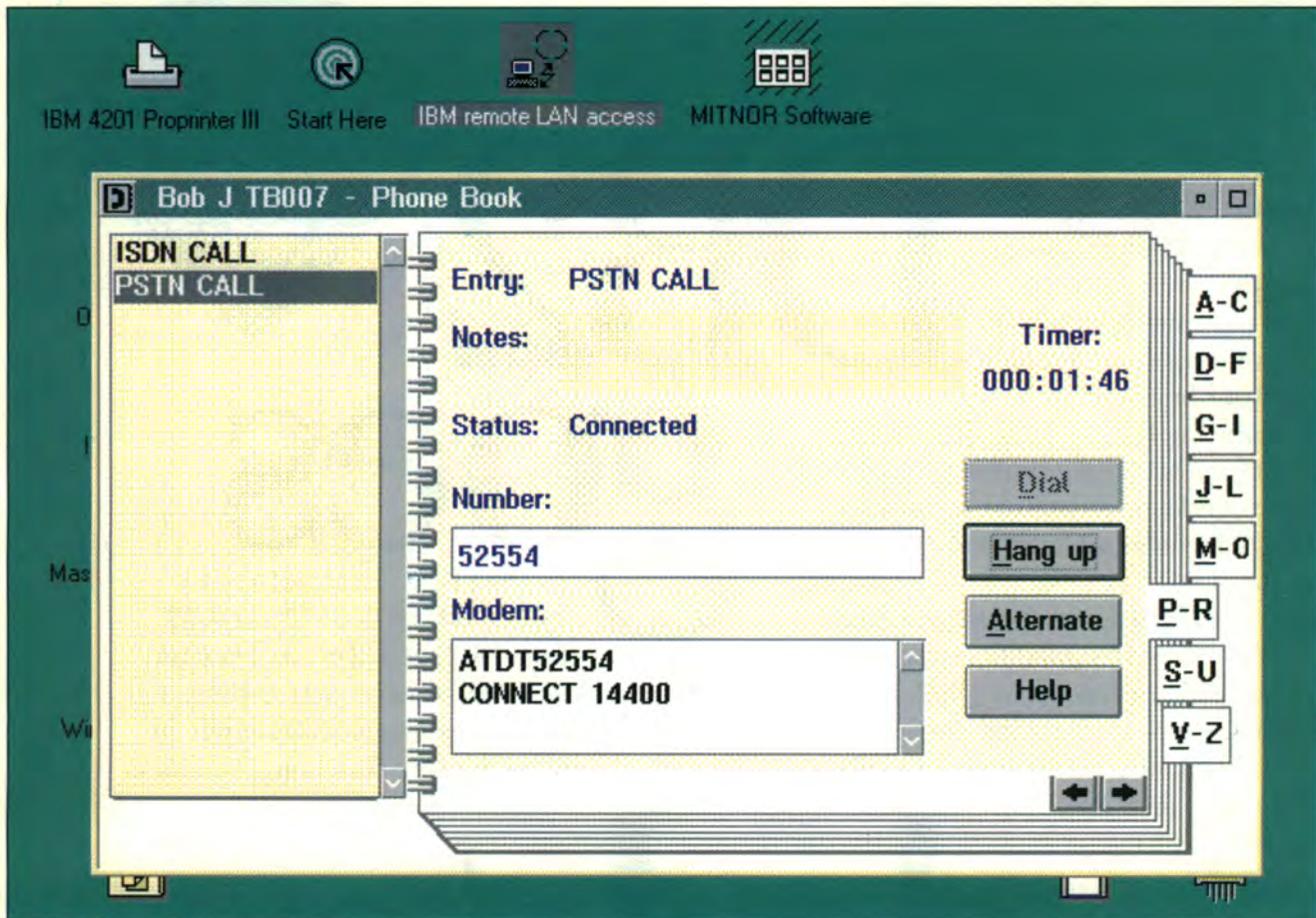


Figure 2. Phonebook and call status

a LAN-attached file server, it can provide indirect remote access to the LAN through shared files contained on the server. This configuration provides the functionality of the remote client approach. If used in conjunction with the Connection Server, a remote workstation can directly access any workstation on the LAN configured to participate in the remote environment. The Remote package runs on either OS/2 2.x or Microsoft Windows 3.1.

The IBM LAN Distance Connection Server provides administrative and security services in addition to routing frames between the WAN and LAN environments. While in small networking environments, the Connection Server can reside on a non-dedicated system. A dedicated system is recommended for supporting large networks with moderate to heavy remote traffic. The Connection Server uses the multi-

tasking capabilities of an OS/2 2.x base.

The LAN Distance product supports all hardware supported by the operating system platform on which the LAN Distance component runs, including IBM and IBM-compatible 80386 and 80486 systems. The product is opti-

settings found there.

When accessing the WAN, a LAN adapter is *not* required on a remote workstation, nor is a modem required on a LAN-attached workstation; communication between the LAN and WAN is accomplished via the connec-

## *LAN Distance consists of two components, the Remote Server and the Connection Server.*

mized for modem and channel speeds of 9600 bps and above. Most commonly used modems are supported via a single selection within the LAN Distance product settings notebook (shown in Figure 1), and many others, such as the Phone Book shown in Figure 2, can be configured by customizing the modem

tion server. Modem sharing is also very cost-effective.

The LAN Distance product includes support for the following connectivities:

- LAN Connectivities—Token Ring, Ethernet
- WAN Connectivities—ISDN Basic



## Client/Server Database Solutions

It's available now—ready to perform on your desktop. A new function-rich, 32-bit relational database you can really trust with your growing client/server network, your mission-critical data *and* your business.

Introducing IBM DATABASE 2™ OS/2® (DB2/2™) from IBM Programming Systems, the birthplace of relational database technology.

DB2/2 includes an industrial-strength DB engine that supports transaction management, concurrency control, security, integrity, and recovery functions. Designed to exploit the power and open architecture of OS/2, it also supports industry-standard SQL for developing portable applications. And it runs your DOS, DOS Windows™ and OS/2 applications requiring *online* access.

You can access data directly from DB2/2 on your desktop or from a DB2/2 server on your LAN, and with

# DB2



# goes

DISTRIBUTED DATABASE  
CONNECTION SERVICES/2,™  
from DB2®, SQL/DS™, and OS/400®

databases as if they were on your desktop, too. This versatility can play a significant role in an Information Warehouse™ solution for your business.

We've developed an

# desktop.



exciting demo diskette to show you just how well new DB2/2 performs—right on your desktop. Call us today for your free demo, or to order DB2/2: 1 800 342-6672; or fax: 1 800 445-2426. In Canada, call 1 800 465-7999, ext. 850. An upgrade from OS/2 Extended Edition or Extended Services is also available.



Rate Adapter, Asynchronous Communications Port, Dual Asynchronous Adapter, Asynchronous/Synchronous RTIC Adapter, Synchronous Wide Area Connector, X.25 via modem.

The LAN Distance product uses IEEE 802.5 source route bridging to support token ring connections and transparent bridging to support Ethernet. Because transparent bridging requires the Ethernet adapters to run in "promiscuous" mode, the number of WAN ports effectively supported on Ethernet will be less than that of token-ring connection servers. This overhead, due to the connection server's increased traffic filtering, is a fraction of that incurred if unfiltered traffic were allowed to flow over the WAN.

The LAN Distance product allows the connection server to use other techniques to move frames to and from the LAN. To offset the processing overhead when using Ethernet or connecting different LAN types, a higher layer router can be used. For example, with a TCP/IP or IPX protocol, the IP router/gateway could be used with the LAN Distance product. The router would then determine which frames would be directed off the LAN to the WAN.

The LAN Distance product's bridging functions provide extensive filtering capabilities, simple network management protocol (SNMP) support, and IEEE 802.1d spanning tree algorithm and protocol (STAP) support. The latter is primarily intended for use in non-switched connection environments.

The LAN Distance product also supports high-speed synchronous non-switched connections via the IBM Wide Area Connector. These include up to two fractional T1 connections or a single T1, E1, or J1 connection.

The LAN Distance product also allows an asynchronous modem equipped with a PAD function to dial into an X.25 network.

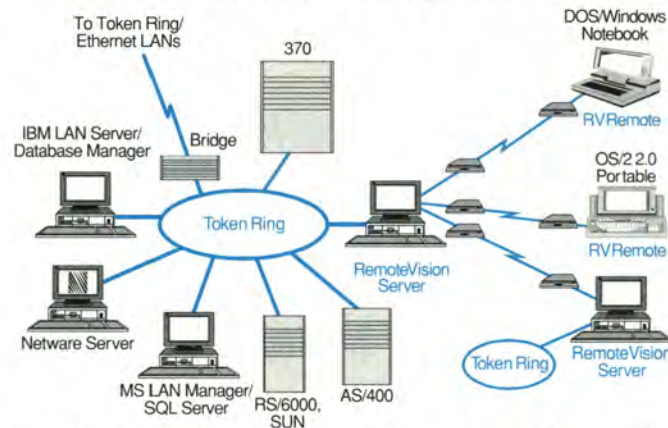
The LAN Distance product includes media access control (MAC) drivers for the first four WAN connectivities. Other adapters packaged with MAC drivers (such as the IBM wide area connector) that adhere to and support the

NDIS interface can also be supported by the LAN Distance product.

The LAN Distance product supports all NDIS-enabled protocols; of those, it contains the following:



## Introducing RemoteVision.<sup>™</sup> The Complete Software Solution For Remote Networking.



- Extend LANs transparently over asynchronous modems to remote workstations and workgroups.
- Run DOS, Windows<sup>™</sup>, & OS/2<sup>™</sup> apps written to multiple protocol stacks on remote nodes.
- Operate remote computers as full-function LAN nodes.
- Connect to remote offices with LAN-to-LAN dial-up networking.
- Add remote connectivity services using existing LAN machines.
- Access remote PC servers and hosts without additional host gateways.
- No specialized hardware/adapters.

RemoteVision makes it easy and affordable for DOS, Windows and OS/2 users in remote locations to connect to Token Ring or other LANs using dial-up modems. Response time remains fast, applications don't need changing, and users won't need retraining. So you can turn your remote machines into full-function workstations. With RemoteVision. To immediately find out more, or to take advantage of our "Try Now, Pay Later" 30-day money-back guaranteed trial offer, call today.



1265 Montecito Ave., Ste. 101, Mountain View, CA 94043  
Tel: (415) 965-8607, Fax info: (415) 965-8607, (then press 2)  
Trademarks are from their respective companies. ©1993 Token Technology, Inc. 001-0





- Netbios
- 802.2
- ODI to NDIS mapper.

Everything required to support these interfaces is included with the LAN Distance product. Users can transparently run any LAN applications utilizing these interfaces within the WAN environment without modification. Applications such as IPX, TCP/IP, Person-to-Person, Lotus Notes, and OS/2 Communication Manager can be purchased and installed to run within the WAN environment. The LAN Distance product can also access an AS/400 via the AS/400 PC Support Program.

The LAN Distance product is network operating system independent

The LAN Distance product's extensive configurable security options, enabled via the settings notebook, include:

- Workstation address identification
- Valid logon time intervals
- Password encryption and session-based user authentication
- Access privilege levels
- Callback.

In addition, the LAN Distance product transparently supports existing LAN and application-level security mechanisms. Security features originating from applications, the network operating system, the operating system platform, and hardware run without modification.

user's location and privilege level are presented.

The graphical user interface provides information on available servers, call status, and hypertext help. Connection to the "virtual LAN" may be made by selecting an entry from a user's phone book, or through a command line interface. Commands may be entered from the keyboard or embedded in a batch or command file.

### **LAN DISTANCE CALLING**

IBM uses the LAN Distance product to provide LAN access to its home terminal users and travelling personnel. It is applied to numerous other tasks, including remote network administration, remote access to mail, field support, video conferencing, remote site data exchange, and remote database and file sharing.

**Pat Scherer**, IBM Corp., 11400 Burnet Rd., Austin, Texas, 78758. Scherer is a developer and performance analyst with IBM LAN Systems. She joined IBM in 1980 as an industrial engineer, developing computer models for strategic planning. Her recent projects include OS/2 Communications Manager and the LAN Distance product. Scherer holds an M.B.A. in industrial management and advanced certification in computer science.

*LAN Distance is used for remote access to e-mail, network administration, field support, video conferencing, data exchange, and remote database and file sharing.*

and is therefore not packaged with any specific network operating system. It is designed to support any network operating system residing over the 802.2, Netbios, ODI, or NDIS interfaces, including:

- IBM LAN Server
- Microsoft LAN Manager
- Novell NetWare Server.

The LAN Distance product employs a standard object-oriented graphical user interface consistent with that of OS/2 2.1. The interface has been designed to be consistent across all supported operating systems and machine types, from Windows-based remote workstations to OS/2-based LAN-attached workstations. Only those selections appropriate to the

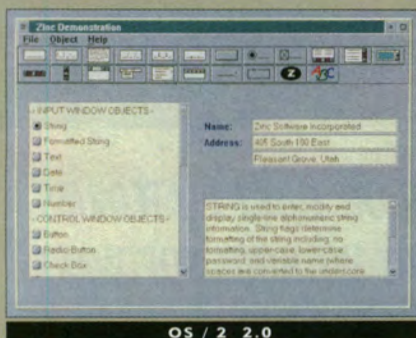


THINK  
ZINC

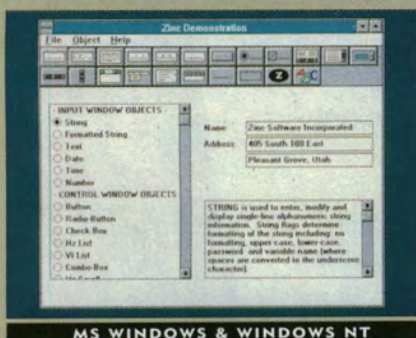
OBJECT-ORIENTED · MULTIPLATFORM · SINGLE-SOURCE · C++



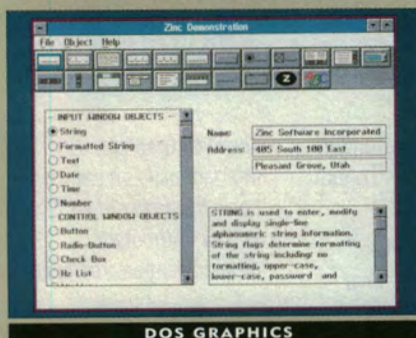
UNIX MOTIF



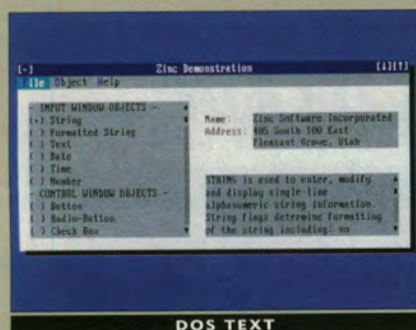
OS / 2 2.0



MS WINDOWS & WINDOWS NT



DOS GRAPHICS



DOS TEXT

## ZINC™ APPLICATION FRAMEWORK™ 3.5

# Multiplatform Flexibility From One Application Framework.

**FLEXIBILITY** is an essential component in software design. Your development tools should allow you to easily incorporate new features and technologies into your applications without rewriting all of your code. That's a tall order if your development tools are designed like a straight-jacket. Zinc Application Framework 3.5 and Zinc Designer™ give you flexible and extendible support for Microsoft Windows, Windows NT, OS/2 2.0, UNIX Motif, DOS Graphics and DOS Text. And Zinc does it with ONE set of source code. Zinc's multiplatform, object-oriented architecture won't confine your application development options today... or tomorrow.

**FOR A FREE ZINC DEMO KIT, CALL US TOLL FREE TODAY AT**

**1.800.638.8665. IN EUROPE CALL +44 (0) 81 855 9918**

Z

i

n

c

ZINC SOFTWARE INCORPORATED, 405 SOUTH 100 EAST, 2ND FLOOR, PLEASANT GROVE, UTAH 84062, TEL 801.785.8900, FAX 801.785.8996, BBS 801.785.8997.

EUROPE: ZINC SOFTWARE (UK) LIMITED 58-60 BERESFORD STREET, LONDON SE18 6BG, TEL +44 (0) 81 855 9918, FAX +44 (0) 81 316 7778, BBS +44 (0) 81 317 2310

© COPYRIGHT 1992 ZINC SOFTWARE INCORPORATED, ALL RIGHTS RESERVED. ZINC, ZINC DESIGNER AND ZINC APPLICATION FRAMEWORK ARE TRADEMARKS OF ZINC SOFTWARE INCORPORATED. OTHER TRADEMARKS ARE OWNED BY THEIR RESPECTIVE COMPANIES.



**PVCS 5.1  
Now Available**

*Jim Besemer*  
Director of Research & Development  
INTERSOLV

**"WE BUILT THREE VERSIONS OF OUR APPLICATION TO RUN ON EIGHT PLATFORMS. THERE'S NO WAY WE COULD HAVE GOT THAT PRODUCT OUT WITHOUT SOFTWARE CONFIGURATION MANAGEMENT. I'D USE PVCS EVEN IF I DIDN'T WORK HERE."**

If you're facing an assignment as tough as Jim's, you're ready for The PVCS Series.

If your development team has more than one member, your application has more than one module or you're working on a LAN—you need automated software configuration management.

**The PVCS Series** is your best choice for LAN-based Software Configuration Management (SCM). We'll help you automate manual processes and avoid the headaches that multi-version, multi-module applications can cause.

You can skip the lost data, the bug fixes that don't stay fixed, the builds that include outdated modules, overwritten code and the frantic midnight phone calls about wrong versions that somehow made their way into production.

**Instead of wasting time** wrestling with your development environment, you can concentrate on building quality applications, regardless of what language, programmer workbench, methodology or operating system you're using.

PVCS eliminates tedious housekeeping details and automates builds so they are consistent and error free.

**PVCS gives you the security** of project-wide undo and the confidence to create good code, knowing you've got a complete audit trail to show where you've been—and help plan where you're going.

Operating seamlessly across Windows, NT, MS-DOS, OS/2 and a variety of UNIX environments, PVCS is flexible enough to fit your development style and the most complex new applications.

**The PVCS Series** is the market leader in SCM for the LAN. It's a complete family of products, designed for the widest variety of distributed client/server development architectures and operating systems.

**The PVCS Series includes:**

PVCS Version Manager, PVCS Configuration Builder, PVCS Production Gateway, PVCS Developer's Toolkit and PVCS Reporter.

**Put the power of The PVCS Series to work for you.** Call for a free demo disk, evaluation, or a copy of the whitepaper, "Software Configuration Management: Choosing The Correct Interface".



*The PVCS Graphical User Interface makes it easy to apply the power of PVCS in your distributed development environment.*

**CALL 800-547-PVCS EXT. 40**

SOFTWARE CONFIGURATION MANAGEMENT FOR HETEROGENEOUS LAN-BASED DEVELOPMENT

**INTERSOLV**

INTERSOLV, Inc. • 1700 NW 167th Place • Beaverton, OR 97006 • 503-645-1150 • FAX 503-645-4576 • E-Mail PVCSINFO@INTERSOLV.COM

1993 © INTERSOLV, Inc. All rights reserved. INTERSOLV, APS, Excelsior, Design Recovery, and PVCS are trademarks of INTERSOLV, Inc. All other products are trademarks and/or registered trademarks of their respective companies.





More than ever, network services are being made available by LANs. NetWare 4.0 for OS/2 can provide network managers with these services from OS/2 2.x. **BY KYLE BIGLER, HEATHER STONE, and BART WISE**

# Using NetWare 4.01 for OS/2

**M**any people implementing complex local area networks (LANs) are finding that more and more network services are being made available via LANs. These requirements continue to put a burden on the network hardware resources. In addition, many developers want to implement client/server solutions in the OS/2 environment that can interact with Novell NetWare file and print resources.

NetWare 3.11 requires that server hardware be dedicated to NetWare and NetWare Loadable Modules (NLMs), applications written to run on a NetWare server. The latest version of NetWare, version 4.01, has an option that gives a new degree of flexibility to NetWare users. NetWare for OS/2 permits the NetWare 4.01 server to be run in a nondedicated fashion. Now any application that runs on OS/2, either 32-bit, 16-bit, or DOS/WIN-OS2 emulation, can share the server hardware with NetWare.

NetWare 4.01 for OS/2 uses the same source code and provides the same functions available in a dedicated NetWare server. NetWare 4.01 for OS/2 contains a set of add-on drivers that permits the native NetWare modules to be loaded in the OS/2 environment. The benefits of running the native NetWare modules in the OS/2 environment are that NetWare NLMs can be run unmodified, NetWare device drivers can be run unmodified, and the high performance and reliability of NetWare is maintained.

## CUSTOMERS

NetWare for OS/2 is targeted at three categories of customers. The first group is customers who currently have strategic applications in both OS/2 or

DOS environments and NetWare.

The second group consists of customers who wish to develop and use OS/2 as a client/server platform. Client/server applications can be developed for OS/2 while maintaining the established file and print server capabilities of NetWare. OS/2 client/server applications can share LAN adapter and disk resources with the NetWare server.

The third group is cost-conscious hardware customers. Customers who are currently using or are planning to use multiple LAN servers can con-

*Now any application that runs on OS/2, either 32-bit, 16-bit, or DOS/WINOS2 emulation, can share server hardware with NetWare.*

solidate all LAN functionality on a single server. The nondedicated operation of NetWare 4.01 for OS/2 will permit other LAN-based applications, such as Lotus Notes, to operate on the same machine. This feature is especially important to customers who have many isolated networks in satellite offices.

## FEATURES

Since NetWare for OS/2 is an add-on to the NetWare package, it provides all the features (except NLM memory protection) provided with NetWare:





- NetWare directory services (NDS)
- New management capabilities
- Multiple language capability
- Enhanced security and auditing
- Routing and WAN improvements
- Data compression, sub-allocation, and data migration
- CD-ROM and optical drive recognition

4.01 for OS/2 server looks the same as a dedicated NetWare 4.01 server.

**Concurrent Use of OS/2.** The preemptive multitasking capabilities of OS/2 allow other workgroup and communication applications, such as OS/2 Communication Manager, OS/2 Database Manager, Lotus Notes, and so on, to run

OS/2-based client/server applications. All NetWare NLMs that have been certified for NetWare 4.01 will run unmodified on NetWare 4.01 for OS/2. NLM development for NetWare 4.01 for OS/2 is supported with the NetWare 4.01 software developers kit. OS/2 client/server applications can access NetWare file and print resource in addition to sharing server hardware resources.

**High Performance.** The NetWare 4.01 for OS/2 server can achieve up to 92% of the performance of a dedicated NetWare server. The small overhead of 8% is due to the fact that OS/2 is now handling all hardware, interrupt, and I/O services. The optimization of these services in OS/2 and the streamlining of the NetWare 4.01 for OS/2 add-on modules results in the small overhead numbers.

**Dynamic Performance Tuning.** This parameter lets you specify what portion of CPU time is available to NetWare; the remaining CPU time is used by OS/2. The performance may be tuned while the server is running by using the graphical monitor utility provided with NetWare 4.01 for OS/2. This parameter may also be set to an initial value via the NET.CFG file.

**Graphical Monitor Utility.** NetWare 4.01 for OS/2 includes a graphical monitor utility that runs as an OS/2 application. This utility monitors the activity of your NetWare server and provides some of the same statistics provided by the MONITOR NLM. The statistics are displayed in graphical and numerical format. You can also use this utility to dynamically adjust performance parameters, memory parameters, and to turn off or on the NetWare server alert beep.

### ARCHITECTURE

NetWare 4.01 for OS/2 runs as a parallel operating system to OS/2. It runs at ring 0 with a protected block of memory dedicated solely to NetWare. NetWare 4.01 for OS/2 comprises, in addition to the Native NetWare 4.01 code,

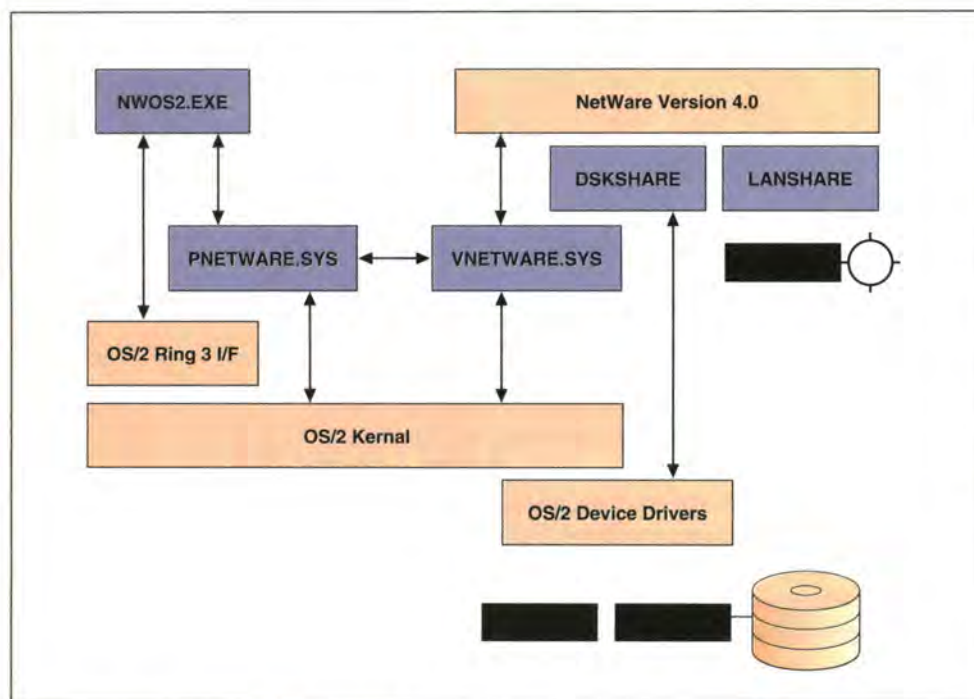


Figure 1. NetWare 4.01 for OS/2 structure

## NetWare 4.01 for OS/2 can be accessed from OS/2, Macintosh, UNIX, DOS, and Windows.

- Graphical administrative tools
- Electronic documentation.

In addition to the features provided by NetWare 4.01, it is possible to realize additional benefits by installing NetWare in the OS/2 environment.

**Accessibility from all Platforms.** NetWare 4.01 for OS/2 can be accessed from the same platforms as NetWare 4.01 (OS/2, Macintosh, UNIX, DOS, and Windows). From a client perspective, the NetWare

concurrently with the NetWare 4.01 server. You may also choose to do work in an OS/2, DOS, or WINOS2 session while the NetWare server provides file and print services for you and other users in your office. This capability gives network administrators the flexibility of managing servers locally or remotely.

**Client/Server Application Development Flexibility.** Application developers have the flexibility to develop NetWare- or



three modules. A virtual device driver (VNETWARE.SYS), a physical device driver (PNETWARE.SYS), and a 32-bit ring 3 OS/2 application (NWOS2.EXE). Figure 1 illustrates the relationship between these three modules, the native NetWare 4.01 code, and OS/2.

NetWare 4.01 for OS/2 also provides the capability to share disk drive, printer, and LAN adapter resources. At the present time, it is not possible for OS/2 and NetWare 4.01 for OS/2 to share an optical drive. Any CD-ROM subsystem must be dedicated to either NetWare or OS/2.

### PNETWARE.SYS

PNETWARE.SYS is an OS/2 physical device driver that communicates with and controls devices through architected OS/2 system calls. This driver has the responsibility for handling interrupts, memory allocation, and screen notification.

### Registering and Unregistering Interrupts.

The system interrupt controller is owned by OS/2, and NetWare must register for the interrupts it intends to use through the architected OS/2 DevHelp routines. PNETWARE.SYS initiates the OS/2 DevHelp routines to register and unregister the interrupts being used by NetWare.

**End of Interrupts.** All first-level interrupt handling is performed by OS/2. When an interrupt occurs, it is first handled by OS/2 and passed to NetWare. All drivers are required to issue an OS/2 DevHelp call to signal that the interrupt controller can be reset to issue another interrupt when one occurs. DOS device drivers handle this by writing directly to the interrupt controller. Since OS/2 owns the interrupt controller, this function must be accomplished using an architected OS/2 DevHelp routine.

**Memory Allocation.** Memory for NetWare 4.01 for OS/2 is allocated by PNETWARE.SYS in one contiguous block. Once this memory is allocated, this block is no longer available to OS/2 or to OS/2 applications. This memory is also locked down and cannot

be swapped out by OS/2. NetWare uses physical addressing; it is crucial that memory for NetWare always be in the same physical location.

The amount of memory to be allocated for NetWare is defined in the NET.CFG file. This memory can optionally be defined so it is released to OS/2 when the NWOS2.EXE process is terminated.

### VNETWARE.SYS

VNETWARE.SYS is an OS/2 virtual device driver that controls NetWare's interface to the server hardware. This module communicates with NWOS2.SYS, PNETWARE.SYS, and OS/2 to perform tasks on behalf of the server. Implementation as a virtual device driver is required to obtain flat 32-bit addressing at ring 0, which is required for the NetWare operating system.

**Interrupt Support.** VNETWARE.SYS contains the interrupt processing routines for the NetWare server. These routines are called by PNETWARE.SYS, which contain the second-level interrupt processing routines for the interrupt levels used by NetWare. Interrupt registration and end-of-interrupt signaling are done in the PNETWARE.SYS module.

**Memory Management.** The block of memory allocated by PNETWARE.SYS is managed by VNETWARE.SYS and NetWare. VNETWARE.SYS is the loader for the NetWare server NLMs. Figure 2 shows how NetWare 4.01 for OS/2 uses memory.

**Screen Management.** The NetWare console screen is managed by communications with NWOS2.EXE. VNETWARE.SYS updates the console and then unblocks an NWOS2.SYS thread. NWOS2.SYS issues the proper OS/2 call to update the screen.

**Keyboard Functionality.** VNETWARE receives information on NetWare console keystrokes from an NWOS2.SYS thread. These keystrokes are then processed as they would be on a native NetWare 4.01 server. The exceptions to

this rule are the Ctrl-ESC and Alt-ESC keystrokes. In NetWare 4.01 for OS/2, you use Ctrl-N and Alt-N to perform the equivalent function. The Ctrl-ESC and Alt-ESC keystrokes have special meanings in OS/2.

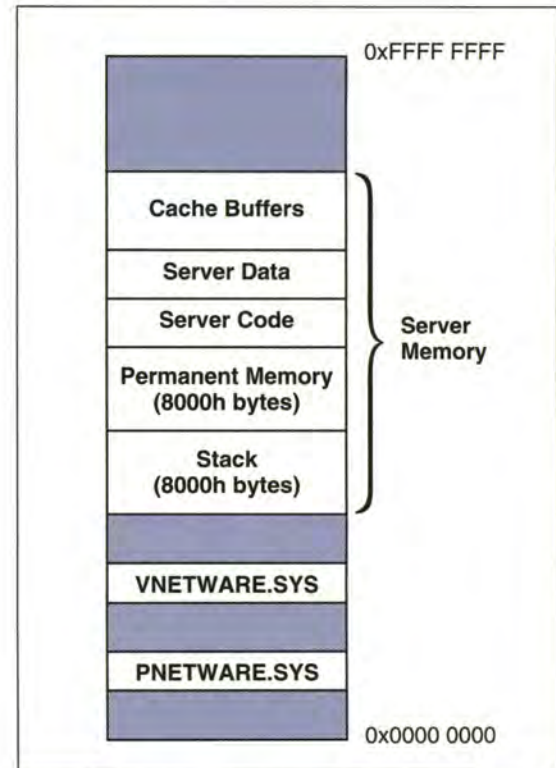


Figure 2. NetWare 4.0 for OS/2 Memory Map

**Timer Support.** All timer processing is done in VNETWARE.SYS.

### NWOS2.EXE

NWOS2.EXE is a 32-bit ring 3 OS/2 application. This module serves as the interface between the user and NetWare. NWOS2.EXE contains four main threads of execution. These threads are dropped down to VNETWARE.SYS and blocked until servicing is required. NWOS2.EXE can be run from either a full screen or windowed OS/2 session.

**Main Thread.** The server's main thread of execution is used to load and run the server. When NWOS2 is entered from the OS/2 command line, this thread is dropped down to VNETWARE.SYS and used as the thread of execution for





the NetWare server.

This thread is blocked if there is no work for the server to do or when the server is busy, according to the performance tuning parameter selected by the user.

**File I/O.** This thread performs local and redirected file accesses on behalf of the NetWare server. All file system I/O must be done at ring 3. VNETWARE.SYS unblocks this thread when OS/2 file I/O is required.

**Screen Handling.** Updates of the NetWare console screen are made via the screen handling thread when the server runs in an OS/2 window. NWOS2.EXE issues the proper OS/2 ring 3 screen-handling function when VNETWARE.SYS unblocks this thread.

**Keyboard.** User keystrokes in the OS/2 session containing the NetWare console are passed down to the server using the NWOS2.EXE keyboard thread.

**Disk Sharing.** DSKSHARE.DSK allows NetWare to use disks controlled by OS/2. This driver provides an interface between NetWare and an OS/2 disk

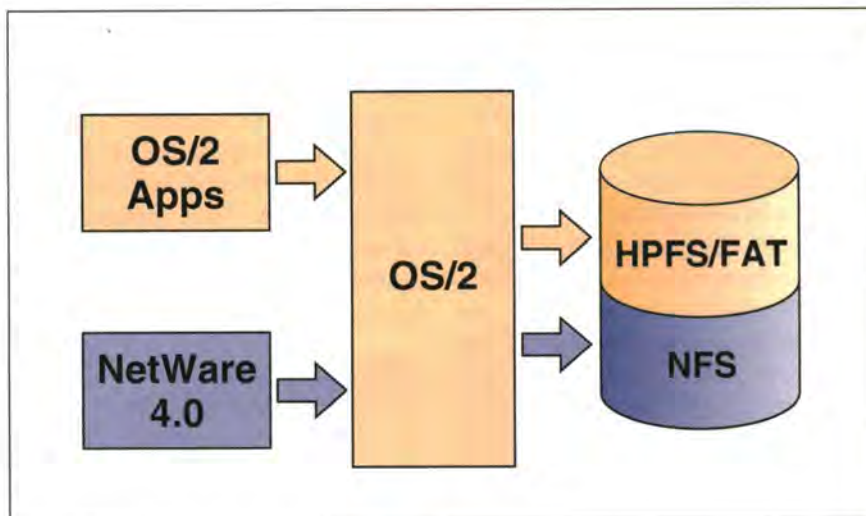


Figure 3. NetWare 4.0 for OS/2 Disk Sharing

and dedicated to either NetWare or OS/2, but they cannot be shared.

For security reasons, the DSKSHARE.DSK driver does not allow transparent access between the NetWare and OS/2 partitions. OS/2 applications can use the space dedicated for OS/2 file systems; NetWare uses the space dedicated to the NetWare file system. OS/2 applications may access the NetWare file system by logging in through a NetWare OS/2 client. This protects your NetWare files from unauthorized use.

by OS/2. Load NPRINT.NLM on the NetWare server to configure the attached printers. When a print job is submitted, NWOS2.EXE will use the OS/2 print queue APIs to manage the print jobs sent by client workstations. From a client perspective, printing is managed exactly the same as on a native NetWare 4.01 server.

**LAN Adapter Sharing.** NetWare 4.01 for OS/2 can share a network board with the NetWare clients on the OS/2 side of the computer using two different methods:

1. Internet protocol (IP) or internetwork packet exchange (IPX) protocols can be sent to the server and other NetWare servers using the LANSHARE.LAN and LANSHARE.SYS modules. This method uses the IP routing function built into the NetWare server.
2. All frame types can be sent to the server side using a virtual token ring driver (TOKENSHR). A bridge on the server, BRIDGE.NLM and VBRIDGE.LAN, can be used to bridge frames to the NetWare server and the external network. This method can only be used if the server has a token ring connection to the external network.

Each of these methods works by set-

*Memory for NetWare 4.0 for OS/2 is allocated by PNETWARE.SYS in one contiguous block. Once this memory is allocated, the block is no longer available to OS/2 or to OS/2 applications.*

device driver. When DSKSHARE.DSK is used, there must be a primary partition on the hard drive that can be dedicated to NetWare. The NetWare partition of the disk is formatted and managed by NetWare. The relationship between DSKSHARE.DSK, NetWare, and OS/2 is shown in Figure 3.

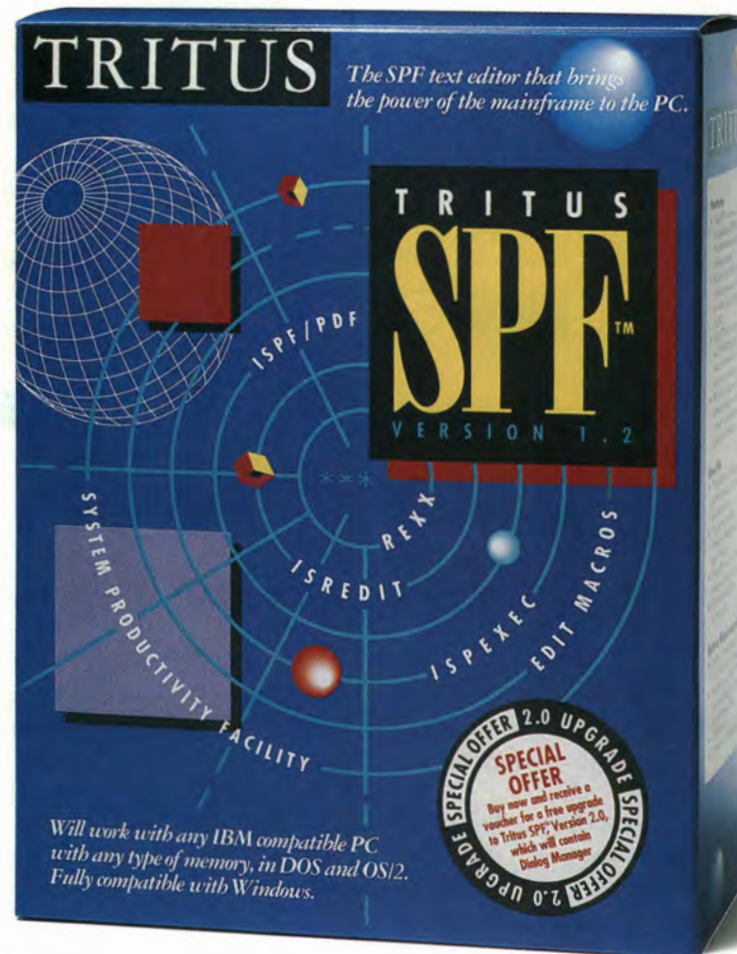
DSKSHARE.DSK is only for shared hard drives. Other disk devices, such as CD-ROM drives, can be installed

Similarly, NetWare clients can only access OS/2 files by logging into the OS/2 distributed application or by physically using the OS/2 computer. This protects the distributed application information.

**Printer Sharing.** Printer services for NetWare 4.01 for OS/2 are provided by OS/2. NetWare for OS/2 supports sharing of all printer types supported



# The only mistake Tritus SPF™ can't undo is your buying another text editor.



**M**ake no mistake about it, only Tritus SPF 1.2.5 has unlimited UNDO/REDO. Tritus SPF is the powerful PC version of the mainframe editor ISPF/PDF and uses the same keystrokes. Features include modifiable panels, keyboard mapping, and integration with Micro Focus Workbench and WorkFrame/2.

REXX edit macros use new optimization techniques for increased performance. You can edit files up to 256 MB in size and 64,000 bytes in width using standard record delimiters, custom delimiters or no delimiters. CUT and PASTE using private or public clipboards (Windows, OS/2)

with append or erase options. A new RECORD mode captures keystrokes automatically and assigns them to a key for playback or future editing. All features, including REXX, work in DOS and OS/2.

Tritus SPF 1.2.5 is available now for \$99 plus shipping and includes a free upgrade to version 2.0 with Dialog Manager. Also included is an OS/2 32-bit version and over 600 pages of documentation. The Prentice-Hall REXX Language book is available for an additional \$25.

We offer a 60-day money back guarantee. To order, call 1-800-321-2100 for same day shipping.

## TRITUS

Tritus, Inc., 6034 West Courtyard Drive, Suite 120 • Austin, Texas 78730-5014 USA  
(512)794-5800, 1-(800)321-2100, Fax (512)794-3833



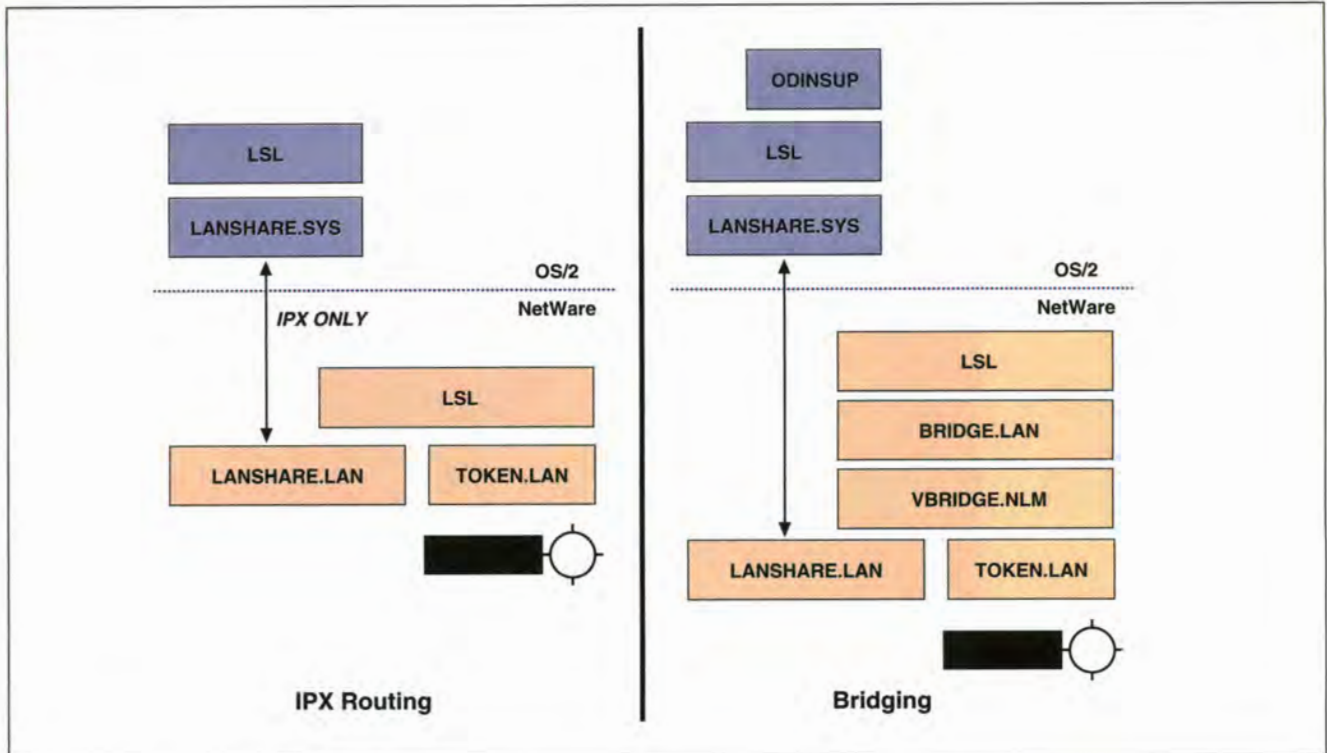


Figure 4. NetWare 4.0 for OS/2 LAN Adapter sharing

ting up a virtual LAN connection on the OS/2 side that communicates with the link support layer (LSL) in the server. Most OS/2 protocols are non-IPX and NDIS (network device interface specification) based and thus require that the TOKENSHR solution be used.

These adapter sharing methods allow an OS/2 NetWare client to operate on the same machine as the Net-

#### INSTALLING NETWARE 4.01 FOR OS/2

NetWare 4.01 for OS/2 is easy to install and requires only a few steps in addition to those required for native NetWare 4.0:

1. Install OS/2 and leave a free space that can be formatted by NetWare. If you have already installed OS/2,

client. This step is required only if you want to use this machine as both a client and a server. Figure 5 shows the sections relevant to NetWare for OS/2 from the CONFIG.SYS and NET.CFG files. In the NET.CFG, the server is specified to use 8MB of memory and have a tuning level of 7. The server will release the memory that was allocated at boot time when the user exits the server.

4. Reboot the computer.
5. Install NetWare 4.01.
6. Define users and manage your network as instructed in the NetWare 4.01 documentation.

*The NetWare 4.0 NWOS2.EXE program is a 32-bit, ring 3 application that serves as the interface between the user and NetWare.*

Ware for OS/2 server. It is possible to login to the server from an OS/2 client on the same machine.

Figure 4 shows the relationship between OS/2 client protocols and drivers to the NetWare 4.01 server ODI drivers.

you can obtain free space by deleting a partition or adding another hard drive.

2. Install and configure the NetWare 4.01 for OS/2 support modules.
3. Install and configure the NetWare

Installing NetWare 4.01 for OS/2 adds about five minutes to the process of installing and setting up a native NetWare 4.01 server. This does not include the time required to install OS/2.





| NET.CFG                                                                                 | CONFIG.SYS                                                                                         |
|-----------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------|
| OS/2 Requester<br>nonded server                                                         | DEVICE=C:\NETWARE\VNETWARE.SYS<br>DEVICE=C:\NETWARE\PNETWARE.SYS<br>DEVICE=C:\NETWARE\LANSHARE.SYS |
| NetWare for OS/2<br>server memory 8192<br>performance tuning 7<br>release server memory |                                                                                                    |

Figure 5. NET.CFG and CONFIG.SYS files

### FUTURE DIRECTIONS

Over time, changes are being planned for NetWare for OS/2 that will enhance its usability and increase the level of

integration with OS/2.

**OS/2 IFS Support.** In the future, any OS/2 installable file system (IFS) will

be mountable to NetWare. This will provide a NetWare client the ability to access any disk partition in the system. This capability can also provide interprocess communication between NetWare NLMs and OS/2 applications.

**Remote Installation Support.** NetWare for OS/2 will be enabled for remote installation. This feature will give network administrators added flexibility in managing NetWare 4.01 for OS/2 servers.

**Memory Protection for NLMs.** NetWare NLMs will be able to run in protected mode identical to the way native NetWare 4.01 has been implemented.

## Universal Document Imaging— An Open Solution

**Magus View™** – The revolutionary new PostScript-language rendering library from Magus. Virtually all business software can generate PostScript output; PostScript is the universal language for the device-independent representation of graphics and text. Now *you* can create programs which display or print documents from any source, without need for the originating application, proprietary file formats, or a PostScript printer.

- Create front ends for document imaging systems – display PostScript files, or use PostScript as the image definition language
- Enhance collaborative applications such as electronic mail or other “groupware” – support documents with complex graphics and fonts
- Create host-based PostScript drivers for non-PostScript printers
- Bring a new level of fidelity to print-previewing in your applications

Magus View is available in DLL form for OS/2 2.0 and Microsoft Windows 3.1. Programming interfaces are provided for C, Smalltalk, and Digital's PARTS Workbench. Prices start at \$495 for a single Magus View Developer's Kit.

**MAGUS**

PO Box 390965 • Mountain View CA 94039-0965 • USA  
(800)848-8037 • (415)940-1109 • fax (415)940-1238 • sales@magus.com

Now Shipping....

## NDP OS/2 2.1 Developer's Pack

**\$595 Includes:**

- 32-bit Globally Optimized Code
- 32-bit Graphics and API Access
- IBM Toolkit and Workframe
- Integrated Development Environment
- 1500 pages of documentation
- Your Choice of Compiler:

**NDP C/C++, NDP Fortran-77  
or NDP Pascal**

Call for our White Papers on OS/2, i860  
Parallel Processing on PCs, Fortran 90 and  
our port of LAPACK to the 486 and i860.

**Microway®**

Research Park, Kingston, MA 02364 USA (508) 746-7341  
U.K., 081-541-5466 USA FAX (508) 746-4678



# Catch

This just in. Applications Manager, best known as *AM*,™ has just added OS/2® 2.1 support. We repeat, the flagship client/server application development software from Intelligent Environments is now available for OS/2 2.1 environments.



For the latest-breaking developments, we take you to corporate America, where *AM* and OS/2 offer a tried and tested mechanism for building 32-bit, multitasking, line-of-business client/server applications. *AM*'s visual programming environment streamlines the development and maintenance of mission-critical client/server applications by teams of programmers. And Static SQL support makes *AM* a real headliner.

This recent news is becoming quite a feature story. By interfacing with MMPM/2, included with OS/2 2.1, *AM* lets programmers employ innovative team development capabilities

# AM

like dynamically linked programming—simplifying reuse and maintenance of program code. DDE support allows programmers to paste information, including *AM* code, comments and *AM*-generated documentation, into Windows™ 3.1 applications easily. And there's also quicker screen interaction and improved productivity through support for OS/2's new high-performance 32-bit graphics engine.

With *AM* and OS/2, you'll never again have to return to your regularly scheduled programming.

# the latest



Use *AM* to develop your own highly rated network programs.

# news.

To order or to find out more about OS/2 2.1 or *AM*, call 1 800 3-IBM-OS2. In Canada, call 1 800 465-7999.

**Operate at a higher level.™**



**Kyle Bigler**, IBM Corp., 11400 Burnet Rd., Austin, Texas, 78758. Bigler is the program manager for NetWare 4.0 for OS/2. He has nine years' experience developing LAN hardware and software. Bigler has a B.S. in electrical engineering from Ohio State University and an M.S. in electrical engineering from Rice University.

**Heather Stone**, Novell Inc., D-21-3, 122 E. 1700 S., Provo, Utah 84606. Stone is the product manager for NetWare 4.01 for OS/2. She has seven years of technical and

marketing experience in the computer industry and has helped develop Novell's OS/2 product line. Stone has a B.A. from Brigham Young University and an M.B.A. from the University of Phoenix.

**Bart Wise**, Novell, Inc., D-21-3, 122 E. 1700 S., Provo, Utah 84606. Wise is a senior software engineer for the Netware for OS/2 product. He has six years experience developing NetWare products. Wise has a B.S. in electrical engineering technology from Brigham Young University.

## REFERENCES

*A Technical Explanation of NetWare 4.01 for OS/2 White Paper* (Novell Inc., March 1993). To order, contact your local Novell or IBM sales office.



## How are you going to test your client/server apps? Check one:

- ☐ Manually
- ☐ Partially
- ☐ The Softbridge  
Automated Test Facility

It's not a trick question, but testing client/server applications can be a tricky business. Traditional, manual methods can't stand up to the complexities of client/server. And most automated testing tools provide only a partial answer — concentrating on the GUI desktop, and leaving the rest of the client/server testing puzzle for you to piece together.

Only the Softbridge Automated Test Facility (ATF) is designed to cover all aspects of client/server testing — GUI, distributed systems, legacy applications. If you're doing client/server development, maybe it's time you checked out ATF. To find out how ATF can help with your client/server testing needs, call 1-800-955-9190.



**The Softbridge  
Automated Test Facility**

 Softbridge, Inc.  
125 CambridgePark Dr.  
Cambridge, MA 02140  
617-576-2257 (Phone)  
617-864-7747 (FAX)





Previous articles in this series discussed issues of control design. This article is the first of two to present advanced control design issues, with a focus on understanding and using color. **BY CHRIS ANDREW, MARK A. BENGE and MATT SMITH**

# A Color-Full Example: Using Color In Control Design



Chris Andrew



Mark A. Bengé



Matt Smith

**O**ur previous articles explained the principles of designing and constructing basic custom controls, illustrated through controls that can be used in everyday applications. Now, with a solid design foundation under our belts, we dive into the advanced control design areas of color, threads, clipping, and memory presentation spaces. We will utilize this control within a Workplace Shell object called a color selector. Source code for the control can be found on the electronic sources listed at the end of this article.

## COLOR THEORY

Before trying to construct a color selection control, we must first understand the basics of color theory as it relates to computer graphics. The following background information is excerpted from *Computer Graphics, Principles and Practice, Second Edition*, by James Foley et. al.

Computer graphics primarily uses five different color models: RGB (red, green, and blue), CMY (cyan, magenta, and yellow), YIQ (U.S. commercial color television broadcasting model), HSV (hue, saturation, and value) and HLS (hue, lightness, and saturation). More information on these color models can be found in *Computer Graphics, Principles and Practice*.

Of the available models, the RGB model is used in OS/2 Presentation Manager and by IBM's and other manufacturers' display and graphics cards. In the RGB model, color is an additive process; colors are made up of a combination of the three available primary colors red, green, and blue. Due

to their additive characteristics, these three colors can combine to make other desired colors.

Despite the use of the RGB model in computer hardware, however, the model we are most inter-

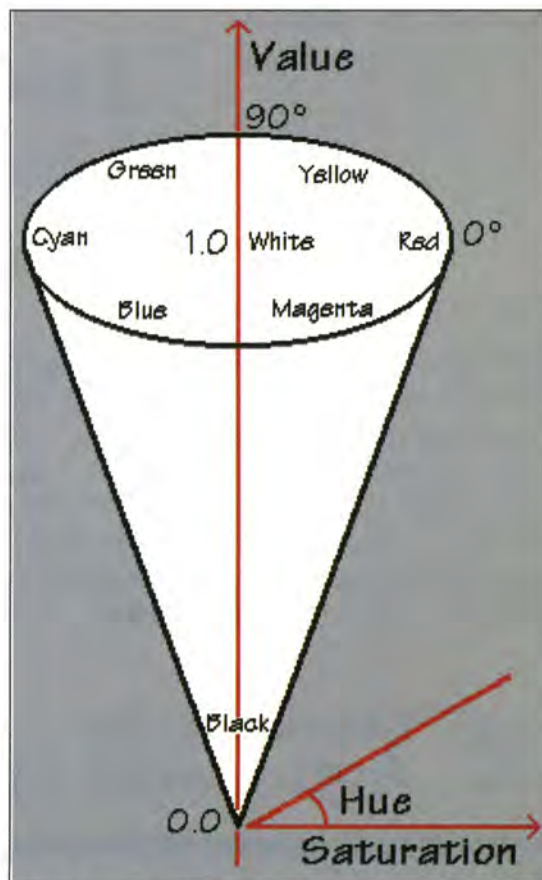


Figure 1. Hue, saturation, and value color model





ested in is HSV, diagrammed in Figure 1. Also known as HSB (B for brightness), HSV can be conceptualized as a spectrum of colors radiating conically from a central axis, as shown in Figure 1. For our purposes, we will take a cross-section of the cone; the remaining portion corresponds to artists' tint, shade, and tone ideals. Since we are using a cross section of the cone, the value component will be at the same level. (In our example, we will use  $V=1$ .)

Foley et. al. describe color theory by assigning a value between 0 and 1 for the saturation (S) and value (V) components. Hue (H) is a value between 0 and 360, corresponding to the degrees of a circle. Blue, for example, is represented as  $H=240$ ,  $S=1$ ,  $V=1$ . For white,  $S=0$  and  $V=1$ ; H can be any value between 0 and 360. When S and V equal 1, they represent the artist's concept of "pure pigment," the starting point for color mixing. When white is added to the mixture, the S value decreases, creating tint. Decreasing the V value while  $S=1$  creates shade. Tones are created when both V and S are decreased.

A selection bar for the value component is provided in the example source code but we will limit the discussion to the wheel itself.

Figure 2 shows the model for our color wheel. It is broken down into six sectors corresponding to the HSV color space. Corresponding colors for Hue are shown at 60° intervals along the wheel.

Our method of calculating corresponding RGB color is derived from Foley's algorithm HSV\_TO\_RGB. To calculate a RGB-based color within Sector 1, the formula in Figure 3 is used.

### GOING FROM THEORY TO PRACTICE

While in the previous model a color's value can range from 0 to 1, within OS/2 Presentation Manager, color is represented on a scale from 0 to 255. We must therefore adapt Foley's algorithms to work within the Presentation Manager's range.

For saturation, which corresponds to the radius of the circle, we will set our range from 0 to 100. For V, we will use a constant value of 1, which leaves a color at its brightest level, allowing no shading.

Using the method for RGB calculation, our color model would be described as shown in Figure 4.

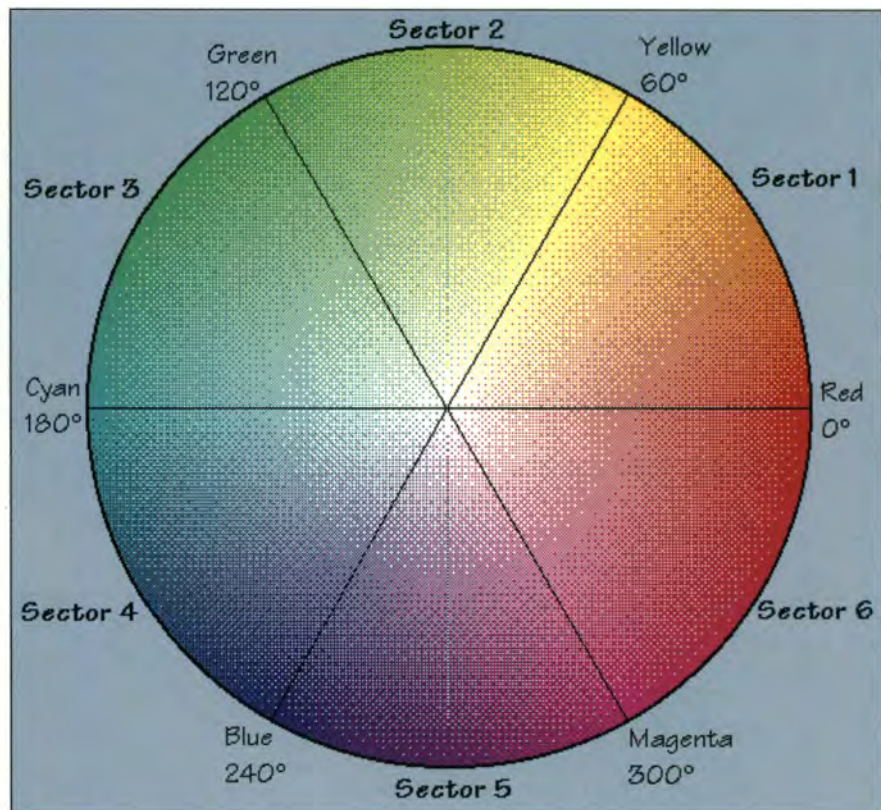


Figure 2. Color wheel model

|       |   |                                                 |
|-------|---|-------------------------------------------------|
| red   | = | 1                                               |
| green | = | $1 - \text{saturation} * (1 - \text{hue} / 60)$ |
| blue  | = | $1 - \text{saturation}$                         |

Figure 3. RGB calculation for Sector 1

|       |   |                                                                                                     |
|-------|---|-----------------------------------------------------------------------------------------------------|
| red   | = | 255                                                                                                 |
| green | = | $((100 - ((\text{saturation} * (100 - ((\text{hue} \% 60L) * 100L) / 60)) / 100L))) * 255L) / 100L$ |
| blue  | = | $((100 - \text{saturation}) * 255L) / 100L$                                                         |

Figure 4. RGB calculation with a saturation range of 0-100, per PM

We must also draw the color wheel using the OS/2 Presentation Manager Gpi APIs. With a solid color rendering such as this, we need to determine



how to draw a given color.

For illustration purposes, let's assume that Hue is in increments of  $10^\circ$  and Saturation is in increments of 25, as depicted in Figure 5. We end up with four drawing areas with the first and last requiring special handling.

We will draw each of the areas in succession, using information from the previous area. The first area, which is shaped differently and does not have a previous area, must be treated as a special case. For drawing each of the line and curve segments, we use four Gpi APIs: `GpiSetArcParams`, `GpiMove`, `GpiPartialArc`, and `GpiLine`. To fill the area with the desired color, we use `GpiSetColor` along with the `GpiBeginArea` and `GpiEndArea` APIs. We will also use `GpiCreateLogColorTable` and `GpiQueryCurrentPosition` to place the presentation space into RGB mode and to determine the current drawing position after drawing the curved segment.

We aren't using the palette management in OS/2 Presentation Manager because not all video hardware has palette support yet. (XGA does, but VGA doesn't.) And the palette documentation in the OS/2 toolkit is woeful, to say the least.

Figure 6 shows line segment and drawing direction. With the Gpi APIs, Area 1 requires no calculation of points. All we have to do is move to the origin of the circle, then draw the partial arc and the closing line segment. In Figure 7, we have shown all the Gpi API calls required to draw Area 1 as if it were a stand-alone figure. In the actual control, we call the `GpiCreateLogColor` table only once, when we enter the routine that draws the color wheel.

After placing the presentation space into RGB mode, we set the color in which the area is to be rendered with the `GpiSetColor` API. We then start the area by issuing the `GpiBeginArea` call. The concept of areas allows us to render the wheel in many colors when we issue the corresponding `GpiEndArea`. (The drawing commands used to describe the area are used to determine the area to be filled with the color specified in

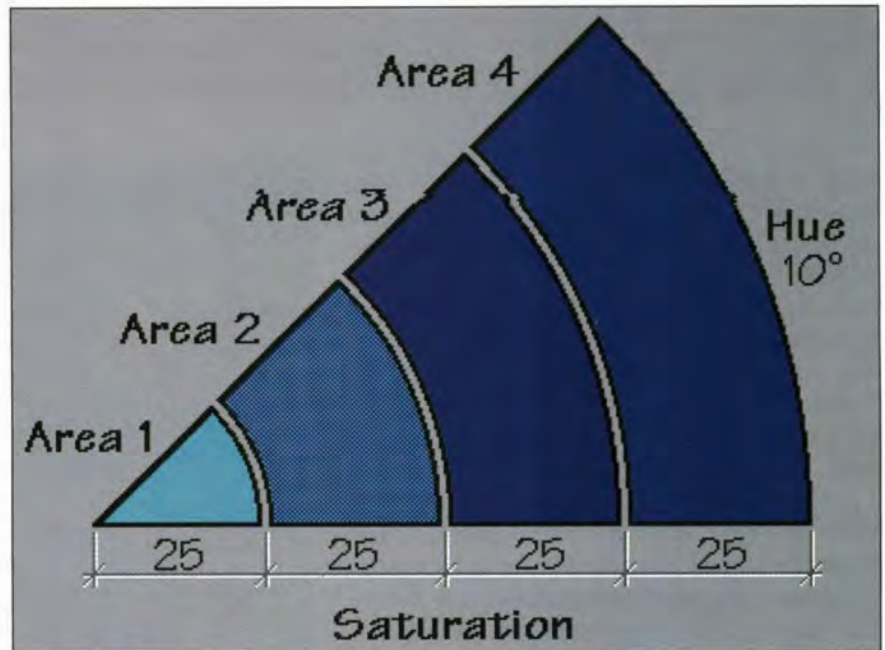


Figure 5. Drawing method

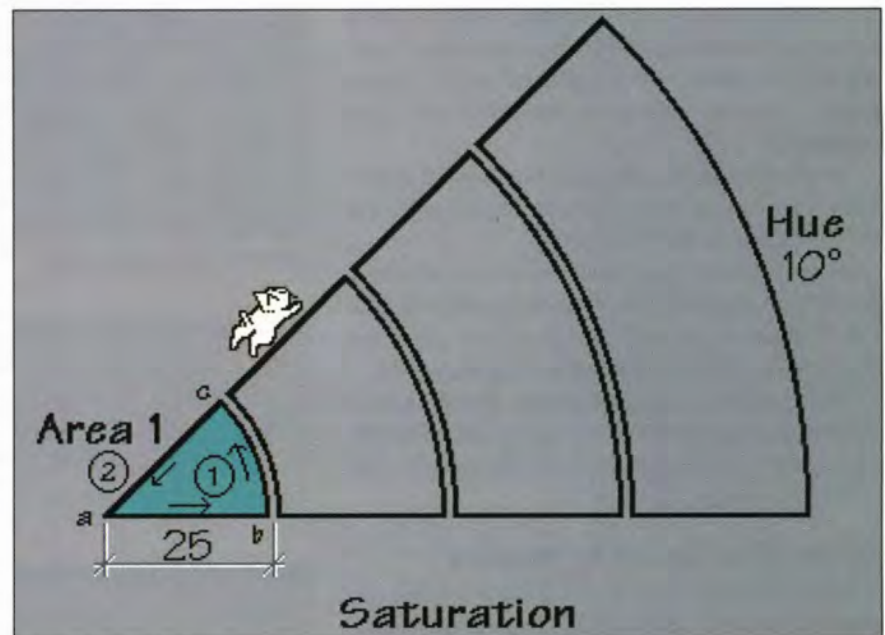


Figure 6. Area 1 drawing method

the `GpiSetColor` API.)

In Figure 7, `GpiPartialArc` draws line segment `ab` for use along with the curved segment `bc`. We then use `GpiLine` to draw the line segment `ca` and issue `GpiEndArea` to render the area.

For Areas 2 and 3, the drawing method is more complex: we must determine points using trigonometry.

Figure 8 shows the drawing method and Figure 9 is a code sample that performs the process.

First, we determine the characteristics of arc `bc` and set them up with `GpiSetArcParams`. We then calculate location `a` with trigonometry, based on the previously drawn arc radius from Area 1. Once `a` is determined, we move to it



# WITT: Instant

Now there's a faster, easier, more affordable way to test your applications. Introducing Workstation Interactive Test Tool (WITT) 2.1, for OS/2® 2.0. An AD/Cycle product from IBM Programming Systems.

WITT does it all. Record, edit, playback and screen compare.

*Introducing  
Workstation  
Interactive  
Test Tool 2.1*

So once you've built a test case, you can use it over and over again.

It works for applications on OS/2 PM, and host-connected MVS, VM, VSE and OS/400®.

Now regression testing is more reliable. New code testing is easier.

• And learning is easier, too. With a friendly online tutorial.

# record,

# instant replay,



What's more, WITT won't test your budget. Because it's very affordable.

It only takes an instant to call 1 800 860-2047, ext. WT1, and ask for more information and our free evaluation demo. Or, better yet, order WITT today with a no-charge two-month testing period and developer hotline assistance.

• The instant you get WITT 2.1, you'll be more productive.

# instant relief.

*The development team at IBM Santa Teresa created WITT 2.1 to help you improve the quality of your applications.*



with `GpiMove` and then cause the line segment `ab` and the curved segment `bc` to be drawn with `GpiPartialArc`. The current drawing position is now `c`; we query it to later draw line segment `dc`. We then move back to point `a` with `GpiMove` and draw curved segment `ad`, this time without a line segment. (This is done by making the arc parameters equal to that of Figure 6 line segment `bc`.) Finally, `dc` is drawn using `c` as the end of the segment `dc`. The area is then closed, forcing color to be rendered inside it.

This method is repeated for each saturation increment up to the last increment value which is similar except for the fact that its radius is the actual radius of the color wheel. This is done because rounding error may occur when the Saturation increments do not divide evenly into 100.

Figure 10 shows the drawing method for the final area. The major difference is that the arc parameters are equal to the radius of the color wheel and need not be calculated.

Now that we understand how the colors within the wheel are drawn, we can begin to explore how to put color into a useable control. Before we lay out the architecture of the final control, however, we will touch on the design goals for our color wheel.

### DESIGN GOALS

Why don't we just use the color wheel within the scheme palette provided with Presentation Manager? Unfortunately, because the color wheel is neither a public method within the Workplace nor a control within Presentation Manager, we can't even try to use it within any of our applications or Workplace objects. As Foley et. al. point out, the HSV method of describing color is also more user friendly, in that the user can see and select from available colors without needing an understanding of color theory. Therefore, our goal is to create a control at least as good as, if not better than, the

```
GpiCreateLogColorTable(hPS, OL, LCOLF_RGB, OL, OL, (PLONG) NULL);
GpiSetColor(hPS, (LONG) rgbClr);
GpiBeginArea(hPS, BA_NOBOUNDARY | BA_ALTERNATE);
ap.LP = ap.LQ = (LSaturationInc * LRadius) / 100L;
GpiSetArcParams(hPS, &ap);

GpiMove(hPS, &ptLOrigin);

/* Draw the outside arc */

GpiPartialArc(hPS, &ptLOrigin, MAKEFIXED(1, 0),
              MAKEFIXED(cAngle, 0),
              MAKEFIXED(LAngle, 0));

/* Draw the closure line from the ending point of the inner arc to the
/* ending point of the outer arc and close the area bracket forcing the
/* color fill of the area drawn in the saturation color. */

GpiLine(hPS, &ptLOrigin);

GpiEndArea(hPS);
```

Figure 7. Area 1 drawing method

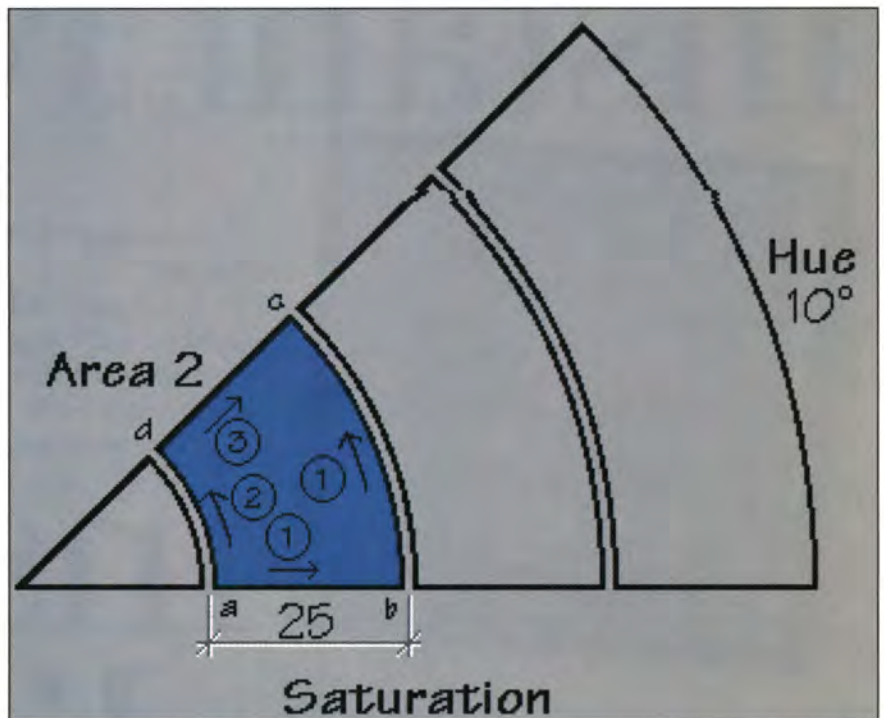


Figure 8. Area 2 drawing method

Workplace color wheel. But "better" comes with a price, as we will illustrate.

### INITIAL DESIGN INVESTIGATIONS

When we were first constructing the color wheel for a real life application,





```
GpiCreateLogColorTable(hPS, 0L, LCOLF_RGB, 0L, 0L, (PLONG)NULL);
GpiSetColor(hPS, (LONG)rgbClr);
GpiBeginArea(hPS, BA_NOBOUNDARY | BA_ALTERNATE);
ap.lP = ap.lQ = (2 * lSaturationInc * lRadius) / 100L;
GpiSetArcParams(hPS, &ap);

ptl.x = (LONG)((double)apPrev.lQ * cos(rdAngle = (double)cAngle /
57.29577951)) + ptlOrigin.x;
ptl.y = (LONG)((double)apPrev.lQ * sin(rdAngle)) + ptlOrigin.y;

GpiMove(hPS, &ptl);

/* Draw the outside arc */

GpiPartialArc(hPS, &ptlOrigin, MAKEFIXED(1, 0),
MAKEFIXED(cAngle, 0), MAKEFIXED(lAngle, 0));

/* Get the ending location of arc to allow for later closure of the area. */

GpiQueryCurrentPosition(hPS, &ptlCurrent);

/* Set the inner arc radius and move back to the calculated starting location. */

GpiSetArcParams(hPS, &ap);
GpiMove(hPS, &ptl);

/* Draw the inner arc */

GpiPartialArc(hPS, &ptlOrigin, MAKEFIXED(1, 0),
MAKEFIXED(cAngle, 0), MAKEFIXED(lAngle, 0));

/* Draw the closure line from the ending point of the inner arc to the */
/* ending point of the outer arc and close the area bracket forcing */
/* the color fill of the area drawn in the saturation color. */

GpiLine(hPS, &ptlCurrent);
apPrev = ap;
GpiEndArea(hPS);
```

Figure 9. Area 2 drawing method

we found that we could, through proper design, allow the specification of the hue and saturation components so the degree of realism that the color wheel displayed was greater than that within the Workplace Scheme Palette.

We found that as we set the hue and saturation to finer values, it took longer and longer to draw the actual color

wheel. Now we had to make a choice: greater realism or better display speed? Table 1 shows times for drawing the color wheel based on the hue and saturation increments.

As you can see, the finer the increments, the longer the drawing time. (The timings shown are based on the final implementation of the color

wheel.) Earlier drawing methods explored included drawing the colors as pie wedges from the origin of the wheel to the outside of the circle, reducing the radius for each increment. This was fine for large increments but could be exceedingly slow with increments of 1° for hue and 1° for saturation. We determined that the drawing routine would have to be as fast as possible and do only necessary drawing.

We began to see some inherent problems when we developed the color wheel as a control, mainly due to the fact that the environment we were using allowed us to treat it like any other control in the system, despite the amount of time it required. Each time the control was resized, or another window that overlapped it was removed, the system would be tied up while the color wheel was redrawn.

We decided to create the wheel as a bitmap in memory to reduce the impact on the system once we had rendered the image in memory. This way, we would have to display the bitmap only when an area of it was invalidated.

Although this solution saved time in later stages, time was still a problem when the control was first created. Since the drawing routine would have created the bitmap within a memory presentation space of the main thread, it retained control of the message queue while the image was being constructed. The only way we could determine to alleviate the problem was to place the drawing within a thread. The result was a design that allowed for the drawing within a thread, and for that drawing to result in a bitmap.

## COLOR WHEEL CONTROL ARCHITECTURE

The resulting control allows for three different drawing methods: the wheel is constructed in real time, as a bitmap image, and as a bitmap image within a secondary thread.

The first method causes the color wheel to be redrawn each time the control is resized or an area of the wheel is invalidated. The second causes the



color wheel to be drawn as a bitmap image, which is displayed each time an area of the wheel is invalidated. (If the wheel is resized, a new bitmap is created for the new size.) In both cases, drawing the color wheel forces a pause in the processing of the message queue since the routine performs all of the drawing before returning control to the Presentation Manager.

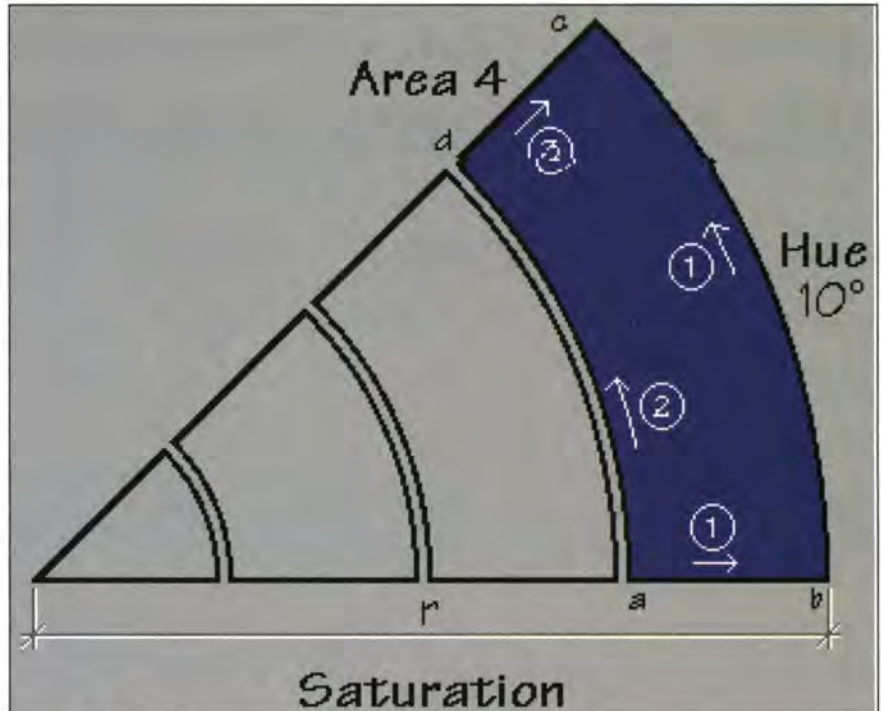
In the third method, based on the second, the drawing is performed in a thread and control is returned to Presentation Manager immediately after the thread is launched.

We designed the color wheel with the styles listed in Table 2 and allowed for a control data structure, as shown in Figure 11. Within this structure, we allowed the specification of hue and saturation increments as well as the radius of the circle.

Instead of describing each of the areas of the color wheel window procedure, we will discuss some areas in which we have implemented special handling for the bitmap and threading.

### **BITMAP CONSTRUCTION**

When creating the bitmap image of the control, we create a memory device context using `DevOpenDC` and then



**Figure 10.** Area 4 drawing method

With `GpiSetBitmap`, we then set the bitmap to use in the drawing and draw the color wheel.

Once the wheel is drawn, we free the bitmap from the memory device context, allowing it to be used within our painting routines. We do this by making the bitmap handle within the

### **THREADING**

In conjunction with the bitmap creation, we allow the bitmap to be created within a thread, thus returning control to the Presentation Manager. This allows the timely creation of all controls within a window or dialogue. If this is not done and hue and saturation values increase by small increments, when you invoke a dialogue your application will appear hung because the dialogue is not immediately displayed. Only when the bitmap image has been created will the processing for the dialogue be complete.

The threading is very simple, but the thread must be coordinated with the rest of the control.

Threading is only used if the control is creating a bitmap image of the color wheel. The thread is launched from two places, the `WM_CREATE` and `WM_SIZE` messages, and is designed to be stopped during the drawing phase since a sizing message can be received by the control before the bitmap image of the color wheel is fully formed.

When the thread routine is started, it issues a `WinInitialize` call. Although this step has never been properly docu-

*When creating the bitmap image of the control, we create a memory device context using `DevOpenDC` and then create a presentation space where the image will be drawn using `GpiCreatePS` for that context.*

create a presentation space where the image will be drawn using `GpiCreatePS` for that context. Once the bitmap format is determined using `DevQueryCaps` and `GpiQueryDeviceBitmapFormats` (i.e., the display type for which the bitmap is being constructed), we create the bitmap using the `GpiCreateBitmap` API.

`GpiSetBitmap` call `NULL`. Finally, we disassociate the presentation space from the device context by specifying a `NULL` value for the device context parameter in `GpiAssociate`, before destroying the presentation space using `GpiDestroyPS` and closing the device context with the `DevCloseDC` API.



# Create GUI Applications That Deliver The Promise Of OS/2.



## YOUR WAIT IS OVER!

Now there's a graphical development and decision support system that combines unparalleled data access and reporting with the stability and multitasking power of OS/2 2.0. It's called PM/FOCUS from Information Builders, a leader in application development tools for almost every environment.

## EXPERIENCE OUR EASE AND SPEED

PM/FOCUS offers a rich graphical toolset for fast, easy application development with a minimum of coding. All application components, including databases, procedures and forms become simple graphical objects that can be

## INTRODUCING PM/FOCUS

controlled by mouse-driven "drag and drop."

Built-in list boxes, check boxes, radio buttons, type fonts and other graphical tools allow you to create attractive GUIs that are so intuitive, they're a snap for even the most unsophisticated end users.

## POWERFUL REPORTING FEATURES

PM/FOCUS offers the most powerful reporting language of any product on the market today. And building reports is a breeze. You simply

transform database fields, record selections, and report headings and footers into selectable objects. Even complex sorts are available at the touch of your mouse.

## DELIVER GREAT OS/2 GUIs

Get the promise of OS/2 now. Make PM/FOCUS your corporate standard for sensational GUI application development. For more information, or to attend a free seminar...

Call 800-969-INFO  
In Canada call 416-364-2760

**IBI PM/FOCUS**  
Information Builders, Inc.





mented in any of the OS/2 toolkits, you must issue a `WinInitialize` call in secondary threads in OS/2 Presentation Manager applications because it prepares the thread to use the `Win*`, `Gpi*`, and `Dev*` API calls. Presentation Manager records information, such as errors, on a per-thread basis through the anchor block.

After the `WinInitialize`, the routine creates a bitmap image of the color wheel. Once the image is complete, the area where it is to be displayed within the control is invalidated with `WinInvalidateRect`. This forces the bitmap image to appear for the first time. Finally, the thread is terminated using the `WinTerminate` and `DosExit` functions.

### COORDINATION ISSUES AND METHODS

Coordinating the drawing thread with the rest of the control initially appears to be a job for semaphores. While they can be used, a simpler method (which has less likelihood of creating a mess with deadlocks) can be implemented using elements of the internal data structure as private signaling mechanisms.

We need to determine the sequence in which the control is to perform its activities. One such activity, stopping the thread while it is

| Hue | Saturation | Time    | Calculations |
|-----|------------|---------|--------------|
| 10  | 10         | 3.8 s   | 3,600        |
| 5   | 5          | 12.4 s  | 7,200        |
| 1   | 1          | 256.3 s | 36,000       |

**Table 1.** Color wheel drawing times (timed on an IBM Model 70-B21, 80486 at 25 MHz)

| Style        | Purpose                                      |
|--------------|----------------------------------------------|
| CWS_BITMAP   | Use bitmap to render image of wheel          |
| CWS_THREADED | Use threading when creating bitmap           |
| CWS_AUTOSIZE | Radius determined by the control             |
| CWS_SOLIDCLR | Use solid colors when rendering              |
| CWS_HSB      | Color notification is HSB CWN_HSBCLRSELECTED |
| CWS_RGB      | Color notification is RGB CWN_RGBCLRSELECTED |

**Table 2.** Color wheel styles

trol indicates that a bitmap image is used to render the color wheel. If this is true, we then check to see if the bitmap is to be created within a separate thread. If this is also true, we then use one of our coordination methods, checking to see if the thread is active. This can be determined because we recorded the thread ID within our private control data; in addition, we have the coordination convention that when a value within the thread ID holder is present the thread is active and when the value is zero the thread is not active.

If a thread ID is present, we issue `GpiSetStopDraw` for the presentation space of the bitmap, with a stop draw condition of `SDW_ON`. Within the drawing routine of the color wheel, we check to see if the drawing is being done within a thread and use `GpiQueryStopDraw` after each hue increment is drawn to see if drawing is to terminate.

Back in sizing processing, after we have issued `GpiSetStopDraw`, the thread is

killed with `DosKillThread` and a new one is launched with `DosCreateThread`.

Once the drawing is complete and the bitmap image has been created, the thread routine clears the thread ID holder and then uses `WinInvalidateRect` to force the bitmap image to be displayed within the control.

To make sure that all of this works properly, the thread ID holder is again used to allow the bitmap image to be displayed within the control. When the `WM_PAINT` message is received, the code checks to see if the style of the color wheel is a threaded bitmap. When it determines this to be the case, it next checks to see if the thread is active by looking for the thread ID within the private control data. If it is active, the outline of the wheel is drawn with the `GpiMove` and `GpiFullArc` APIs. If it is no longer active, however, the code is bypassed and a check is made for the bitmap handle. If the handle is present, the bitmap is rendered with `GpiWBitBlt`.

*A threaded bitmap allows the timely creation of all controls within a window or dialogue.*

drawing the bitmap, occurs when the window is being resized before another drawing request completes. This is done with the `GpiSetStopDraw` and `GpiQueryStopDraw` APIs, specifically designed for drawing in threads. When we receive the `WM_SIZE` message, we check to see if the style of the con-



# OS/2 PRODUCTS & SERVICES

## Consulting **OS/2 PM CUA**

Programming  
Management | Testing  
Analysis | Installation  
Design | Maintenance  
Real-Time Controls and device drivers  
**(708) 380-3584**

## **BUILD YOUR OWN OS/2 LIBRARY**

Reprints are now available for **OS/2 DEVELOPER**.

Use your customized reprints for: **Customer support, sales tools, marketing support, educational tool.**

Call today for a custom quote: Laura Pullen, **OS/2 DEVELOPER**  
415-358-0126 ext.302 or fax 415-358-9749

## **SUBSCRIBE TODAY!**

Only \$39.95 for a full year subscription to the premier source of tools and techniques for the OS/2 software developer: **OS/2 DEVELOPER**

Call today and don't miss a single issue:

**1-800-WANT-OS2**

## **OCR for OS/2**

The most accurate and fastest Optical Character Recognition (OCR) software engine on the market today.

OCR Library and API Toolkits available for OS/2, MS Windows, Macintosh and UNIX.

### **Cognitive Technology Corp.**

Larkspur, CA  
Ph. (415) 925-2323 - Fax (415) 461-4010

## **TCP/2™** tcp/ip for OS/2

"TCP/2™ is, by far, the best implementation of the tcp/ip application suite for DOS and OS/2 available from any source"

Microsoft ITIS

### **Essex Systems, Inc.**

One Central Street Phone (508) 750-6200  
Middleton, MA 01949 Fax (508) 750-4699



Custom Entryfields allow formatting for fields like Phone Number, Dates, Social Security Number, Drivers License, etc.

Easy to use. Just change the window class in your resource file.

## **ENTRYFIELD PICTURE MASKS for OS/2**

Download free demo of Impact Entryfields from our BBS.  
(818) 879-7405

Add Impact to your application

To order call or write  
(800) 676-9390  
(818) 879-5592  
FAX (818) 879-5593



**IMPACT SOFTWARE**

5889 Kanan Rd.  
Suite 330  
Agoura Hills, CA 91301



## **Back In A Flash!**

File Backup and Archiving Facility

- ♦ Versatile, 32-bit, multithreaded file backup and archiving facility for OS/2 2.1.
- ♦ Easy-to-use, PM-based utility supports HPFS, timed/scheduled backups, and file compression.

### **To order, call or write:**

(612)339-5870 ♦ Fax: (612)339-5965  
CCT Inc. ♦ 111 Third Avenue South ♦ Minneapolis, MN 55401  
(Visa, Mastercard, Cash, Check, PO)

Download  
**FREE**  
Demo from  
our 24-hour  
BBS:  
(612) 339-3245  
(8,N,1)

## *Schedule Programs and Reminders ...*



## **Chron v3.0**

Event Scheduler for OS/2

**Only \$69**



**Hilbert Computing**  
1022 N. Cooper  
Olathe, KS 66061

Voice: (913) 780-5051  
BBS/Fax: (913) 829-2450  
CIS: 73457,365



## CLIPPING

You may be wondering what clipping has to do with a color wheel and why we would want to use it. Quite simply, we need to use clipping because we never liked doing trig in school!

Once the color wheel is displayed, it is used to select colors. The easiest way to implement color selection is to let the user move the mouse pointer over the desired color and click. But what about

```
typedef struct _CLRWHLCDATA
{
    ULONG    cb;                /* Structure Size */
    LONG     lAngle;            /* Hue */
    LONG     lSaturationInc;     /* Saturation Increment */
    LONG     lRadius;           /* Radius */
} CLRWHLCDATA ;
```

Figure 11. Color wheel control data

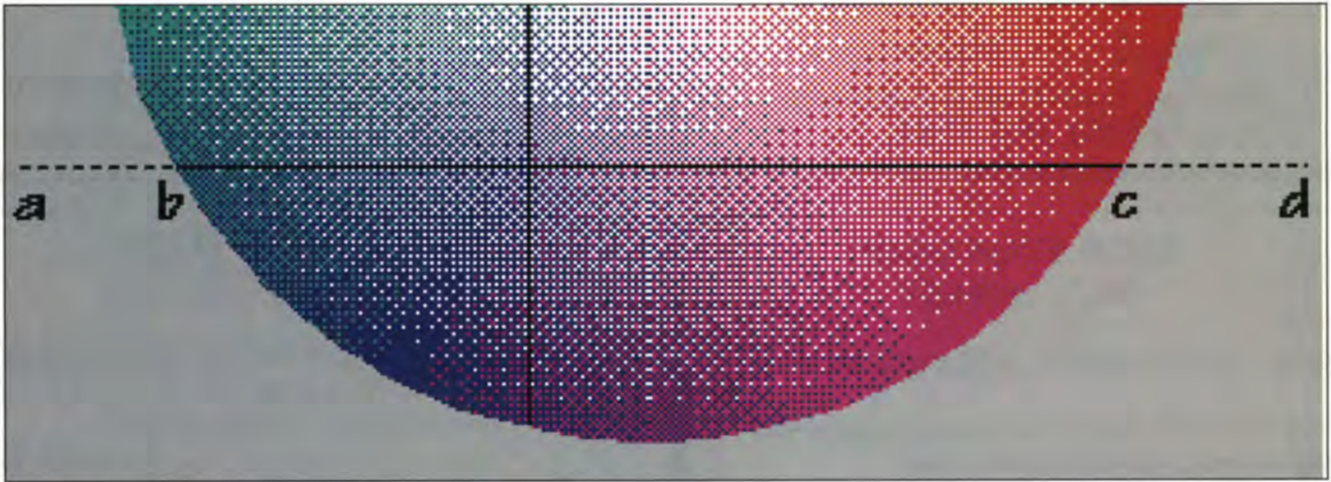


Figure 12. Crosshair clipping

informing the user visually of the location after it is selected? This is where clipping comes in.

The best way to depict the location is to use an exaggerated crosshair, always confined to the wheel, transversing the circle horizontally and ver-

cross will not stop at the intersection point of the circle but may fall outside or just inside the circle.

The second creation method involves clipping. Here, we describe the outline of the circle through Gpi APIs and let the Gpi clipping routines

point of the circle with GpiMove, we draw the circle with GpiFullArc and close the path with GpiEndPath. Having described the circle, we invoke clipping to the path described by the circle with GpiSetClipPath. Not that hard from our point of view, since all the hard work is done by the Gpi APIs. We can now draw the crosshairs using the normal GpiMove and GpiLine APIs. The starting and ending points can be outside the limits of the circle, as shown in Figure 12. We draw the line and Gpi clips the line segments ab and cd, showing only the segment bc.

## SELECTION

Clipping is also used to determine if the user has clicked the mouse pointer within the color wheel circle or if he or she is dragging the crosshair inside it. Here, we use clipping to determine if the mouse point is within or outside the circle. Again, we must describe the circle as we did for the crosshairs; instead of drawing the crosshairs, how-

*Clipping is also used to determine if the user has clicked the mouse pointer within the color wheel circle or is dragging the crosshair inside it.*

tically. There are two ways to create the crosshair. One is to calculate intersection points on the edge of the circle and draw from those points. This method tends to be a bit on the slow side since we have to use trigonometry to calculate the points. Also, if any rounding off occurs within our calculations, the

clip the cross hair to the circle limits for us, even if we start and end the crosshair lines outside the limits of the circle, as shown in Figure 12.

To use clipping, we first define the characteristics of the circle with GpiSetArcParams and start a path with GpiBeginPath. After moving to the center



ever, we check the point where the mouse was clicked or moved to with `GpPtVisible`. If `GpPtVisible` returns a `TRUE`, the point is within the circle. If it is `FALSE`, it is outside the circle.

### STILL TO COME

In a future article, we will show how to integrate the color selection control with a `WorkPlace` object.

**Chris Andrew**, *WordPerfect Corp.*, 1555 N. Technology Way, Orem, UT 84057. Andrew works in the OS/2 Shared Code group at WordPerfect Corp. He previously worked for IBM Hursley (U.K.) and IBM Boca Raton. He has been involved with OS/2 Presentation Manager since the OS/2 1.2 release and most recently was a member of the OS/2 Workplace Shell team on the 2.0 release. Andrew has an honors degree in physics from Imperial College of Science and Technology, London.

**Mark A. Benge**, *IBM Programming System Laboratory*, 11000 Regency Pkwy., Cary, NC 27512. Benge is a staff programmer who joined IBM in 1989 and has since worked on various CUA '91 controls for OS/2 2.x and CUA Controls Library/2. He now works in IBM class library user interface development, where he is involved in the implementation of C++ classes for direct manipulation. Benge received a B.S. in computer science from Western Carolina University.

**Matt Smith**, *Prominare Inc.*, 100 Front St. E., Toronto, Ont., Canada M5A 1E1. Smith is lead architect for the Prominare Development System, an OS/2 2.x advanced GUI development environment. He has been actively involved with OS/2 since 1988 and co-founded Prominare in 1990. Smith has a degree in architecture from the University of Waterloo.

## REFERENCES

Benge, Mark and Matt Smith, "Demystifying Custom Controls," *IBM OS/2 Developer*, Winter 1993: 120-131.

Benge and Smith, "Designing Custom Controls," *IBM OS/2 Developer*, Spring 1993: 72-85.

Foley, James, Andreis van Dam, Steven Feiner, and John Hughes, *Computer Graphics Principles and Practice, Second Edition*. Reading, Mass.: Addison-Wesley System Programming Series, 1990.

*OS/2 2.0 Programming Guide Volume II* (IBM Doc. S10G-6494)

*OS/2 2.0 Presentation Manager Programming Reference Volume I* (IBM Doc. S10G-6264)

*OS/2 2.0 Presentation Manager Programming Reference Volume II* (IBM Doc. S10G-6265)

*OS/2 2.0 Presentation Manager Programming Reference Volume III* (IBM Doc. S10G-6272)

The source code can be downloaded from several electronic sources:

- In the U.S., call the IBM PCC BBS at (404) 835-6600. The source code for this article is in File Area 11 under the name CTRLDES.ZIP.
- In Europe, you can find the source code on the International OS/2 Users Group BBS at +44 (0)454 633197 or +44 (0)454 633420.
- On CompuServe, the source code is in LIB13 of the OS2DF2 forum.
- If you have a VM account on an IBM node, you can receive the source code with the command `REQUEST CLRWHL FROM BANZAI AT CARVM3`.





# FINALLY, CLIENT/SERVER INTEGRATION.

Not long ago, client/server development required massive amounts of time, money and expertise to combine different and complex technologies.

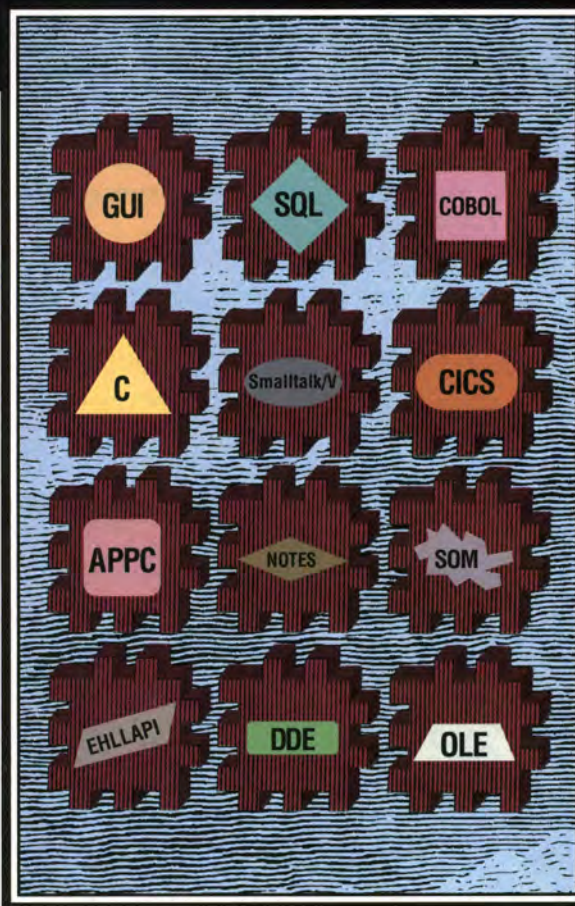


Now Digitalk PARTS®, a rapid application development tool set, lets you easily integrate your software assets into client/server applications.

PARTS is the only object-oriented technology that lets you leverage your legacy code and the knowledge of your current staff.

Only PARTS products let you take existing code—written in Smalltalk/V, COBOL, C, SQL and other languages—and wrap it into components or “parts.” Which can then be virtually snapped together visually. The result is smooth-running client/server applications in a fraction of the usual time. For a fraction of the usual cost.

PARTS supports all popular SQL databases like Sybase, Oracle and DB2. Plus legacy or late model



systems like CICS, COBOL, APPC and SOM. And PARTS lets you develop on both OS/2 and Windows.

**RATED #1 – TWICE.**

Only months ago, PC WEEK awarded PARTS Workbench the highest rating ever in the OS/2

category, calling it “the definitive visual development tool.”

And InfoWorld ranked PARTS the #1 component-based tool for visual development. InfoWorld's Stewart Alsop adds: “There’s nothing like it on the PC.”

To make large teams productive, PARTS also supports group development and version control. Plus PARTS has a host of graphical power tools to give you all the power of objects—without the learning curve.

**10 YEARS EXPERIENCE.**

And PARTS is from Digitalk. The company that’s been providing object-oriented tools to the Fortune 500 longer than anyone else in the world—with over 125,000 users.

Call 800-531-2344 X 606 and ask about our

PARTS Workbench Evaluation Kit.

With minimum effort, you’ll learn why PARTS is the maximum solution for client/server integration.



**PARTS. THE CLIENT/SERVER INTEGRATION TOOL.**

**DIGITALTALK**





The IBM Graphics Interface Kit/2 helps programmers develop intelligent graphical interfaces for applications. This article explores the main concepts of GIK/2 and explains how to use the product to develop an application interface. **BY ROLAND HATSCHKA and SILVIA HANSJAKOB**

# Graphics Interface Kit/2: Visualizing and Manipulating Data

**D**esigning and implementing intelligent graphical interfaces is a complex task requiring a substantial investment of time, money, and skilled personnel. Changing requirements during development can necessitate significant changes to an application's interface. Graphics Interface Kit/2 lets developers quickly produce and modify intelligent graphics interfaces, making it easier to reallocate valuable resources and reduce development time.

Graphics Interface Kit/2 (GIK/2) minimizes the amount of code that must be written to prototype and develop intelligent graphical interfaces for applications. GIK/2's customizable, intuitive interface and easy-to-use design tool let developers create prototypes of interfaces in one or two hours. The product can also generate the C source code needed to link the interface with the application code. A library of almost 300 high-level functions helps developers further refine interface behavior, and a set of layout algorithms allows automatic positioning of the objects on the screen.

## CONCEPTS

There are two classes of symbols in GIK/2, nodes and links. GIK/2 interprets displayed symbols as a network diagram. The symbols, used to map user actions to application functions, are manipulated to access and change application data.

**Nodes.** Nodes usually represent data objects (for example, airplanes or people, as shown in Figure 1, or processes or data stores). Each node in a window can be independently positioned and manipulated.

**Links.** Links usually represent relationships (for example, wires, business reporting lines, data flows, or control flows, as shown in Figure 2, or wires or business reporting lines). They are connected to nodes or other links and are affected by actions performed on them.

GIK/2 provides a WYSIWYG editor for creating symbols. Symbols can have more than one part so as to reflect changes to underlying application data. For each symbol part, you can specify:

- What the symbol part looks like
- What happens when a user clicks on the symbol part
- Which application data or functions are represented by the symbol part.

At least one form must be defined for each symbol. While semantic variants for a symbol can be shown by defining more than one form, the code used to manipulate the symbol needn't be changed. To show the state of a symbol, multiple shapes can be defined for each form. For example, you may want to specify a specific shape when a symbol is dragged across the window. For each form, up to four shapes can be defined to determine a symbol's look in the following locations and states:

- In the main window (standard shape)
- In the palette (palette shape)
- While being dragged (drag shape)
- While being attached to another symbol (attachment shape).



Roland Hatschka



Silvia Hansjakob



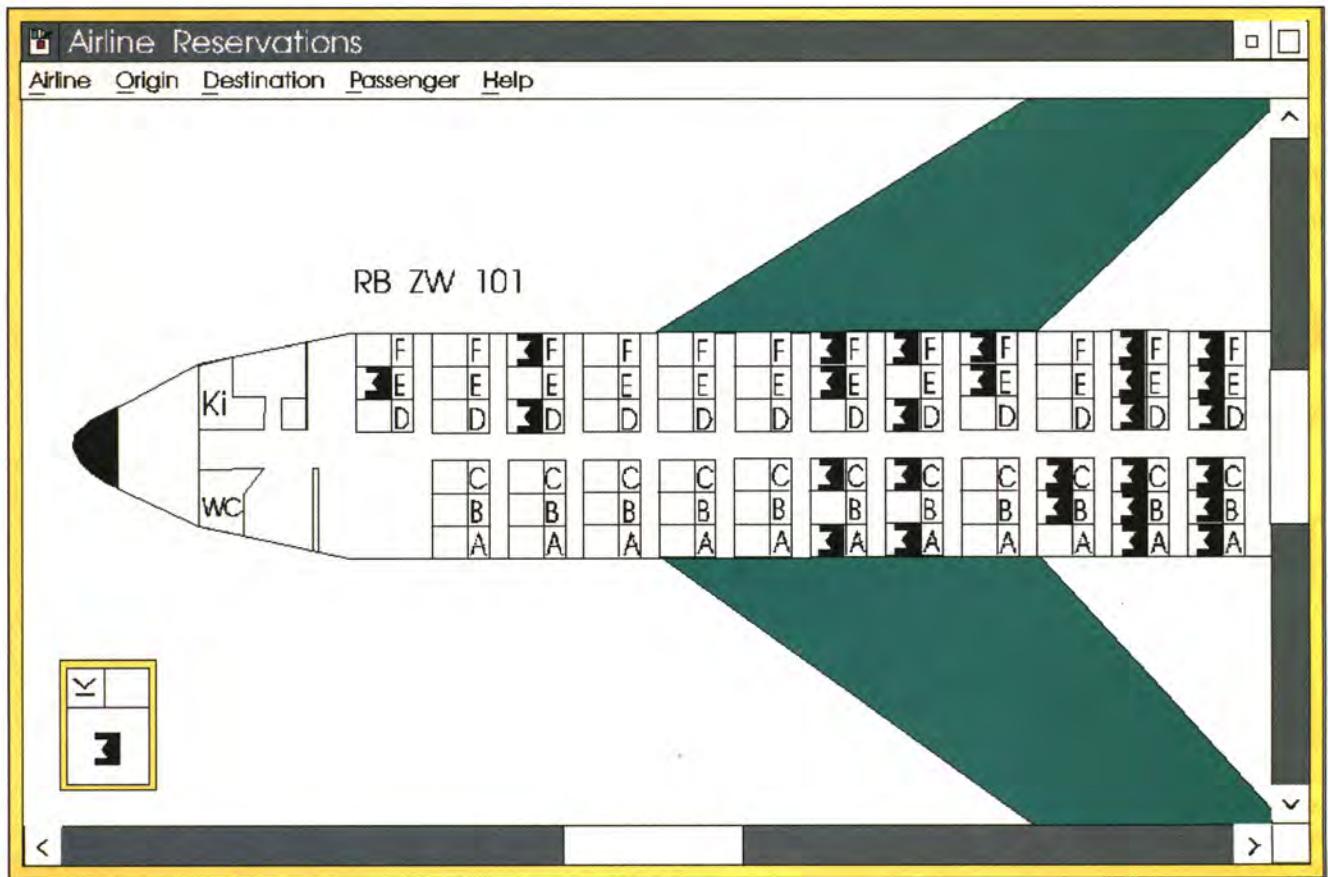


Figure 1. Airline reservations system built with Graphics Interface Kit/2

### DEVELOPMENT OVERVIEW

The development process consists of several steps. You first prototype an interface using the GIK/2 design tool, an interactive tool with a WYSIWYG drawing facility that can be used by

When the prototype is satisfactory, it is time to write customizing code (C/C++ functions). The code links the interface with the application data and implements any special menu choices created for users.

Help for users is typically the last development step. Here, GIK/2 provides a library of generic help panels that can be edited to describe the unique features of an interface.

### THE GRAPHICS EDITOR

The graphics editor can be thought of as a generic interface; a set of windows, menus, and supporting features often needed in a graphical interface. At run time, this generic interface can be dynamically modified by:

- The design file, which populates the interface with symbols
- The customizing code, which synchronizes the interface with the application and implements any special menu choices that have been created
- The help library, which provides users with context-sensitive information.

*GIK/2 minimizes the code that must be written to prototype and develop intelligent graphical interfaces for applications.*

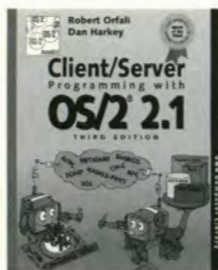
nonprogrammers. The prototype is then saved in a design file that can be repeatedly tested, refined, and extended. Initially, the prototype need contain only an application-specific symbol set and a palette from which instances of the symbols can be dragged.

GIK/2 supports these programming tasks in two ways. First, it generates source code for the developer's customizing code. Second, GIK/2's library of about 300 high-level functions can be called from either the customizing code or the application main code.



# MASTER RESOURCES FOR OS/2 PROS ON THE FAST TRACK

## VNR—YOUR OS/2 POWER SOURCE



Chosen "OS/2 Book of the Year"  
by OS/2 MONTHLY!

*New Edition...Just Released!*

### CLIENT/SERVER PROGRAMMING WITH OS/2® 2.1

**Third Edition**

By Robert Orfali and Daniel Harkey

The ultimate book for anyone planning a client/server environment. Readers raved about the first two bestselling editions—now the third edition covers all the demands of 2.1 and gives proven examples tested in 2.1. Learn how to create client/server applications, including multitasking, LAN communications, the Database Manager, transaction servers, and more!

Praise for previous editions:

"...a must for corporate developers."  
—WILL ZACHMANN, PRESIDENT, CANOPUS RESEARCH.

"...brings all the pieces together."  
—IBM SYSTEMS JOURNAL

**\$39.95, 1,026 pages, paper,**  
**ISBN 0-442-01833-9 (October)**



*New!*

### USING WORKPLACE OS/2®

**Power User's Guide to IBM's New  
Operating System/2 Version 2.1**

By Lori T. Brown and Jeff Howard

Get the "inside" help of the Workplace Shell's lead designers to convert easily and quickly from Windows and Mac environments to OS/2 • set up, maintain, and customize your own Workplace environment • power up with OS/2's Multimedia Presentation Manager/2 under Workplace.

**\$24.95, 464 pages, paper with disk,**  
**ISBN 0-442-01590-9**



*Just Released!*

### OS/2® 2.1 REXX HANDBOOK

**Basics, Applications, and Tips**

By Hallett German

Use REXX in OS/2 and across other platforms...this "one-stop" source gives you everything you need to • develop a REXX application • interface REXX with third-party editors, programming utilities, and languages • choose REXX interpreter benchmarks • interface REXX with Database Manager, Communications Manager, and other IBM extensions • debug REXX procedures • use MacroSpace operations and the variable pool interface • and much more!

**\$29.95, paper, 432 pages,**  
**ISBN 0-442-01734-0**

#### ALSO AVAILABLE:

#### OBJECT-ORIENTED SOFTWARE DEVELOPMENT

Engineering Software for Reuse

By John McGregor and  
David Sykes

**\$42.95, paper, 0-442-00157-6**

#### THE COBOL PRESENTATION MANAGER PROGRAMMING GUIDE

By David M. Dill

**\$39.95, paper, 0-442-01293-4**

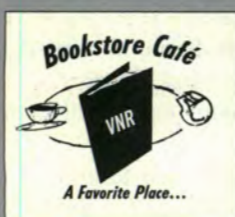
#### COMPREHENSIVE DATABASE PERFORMANCE FOR OS/2® 2.0'S EXTENDED SERVICES

By Bruce Tate, Terry Gray,  
and Tim Malkemus

**\$39.95, paper, 0-442-01325-6**

Now open on the CompuServe® Network:

## THE VNR BOOKSTORE CAFÉ—



*Your hotline to the best solutions on-line.*

Just GO VNR and shop at your leisure from a vast selection of guides and software tailored to your needs. Convenient on-line ordering!

**EASY ORDERING! Call TOLL-FREE:**  
**1-800-544-0550 (Dept Z 1537)**  
**(and get a full listing of VNR titles).**  
**Or Fax 1-606-525-7778.**



**VAN NOSTRAND REINHOLD**  
Publishing for Professionals Since 1848.



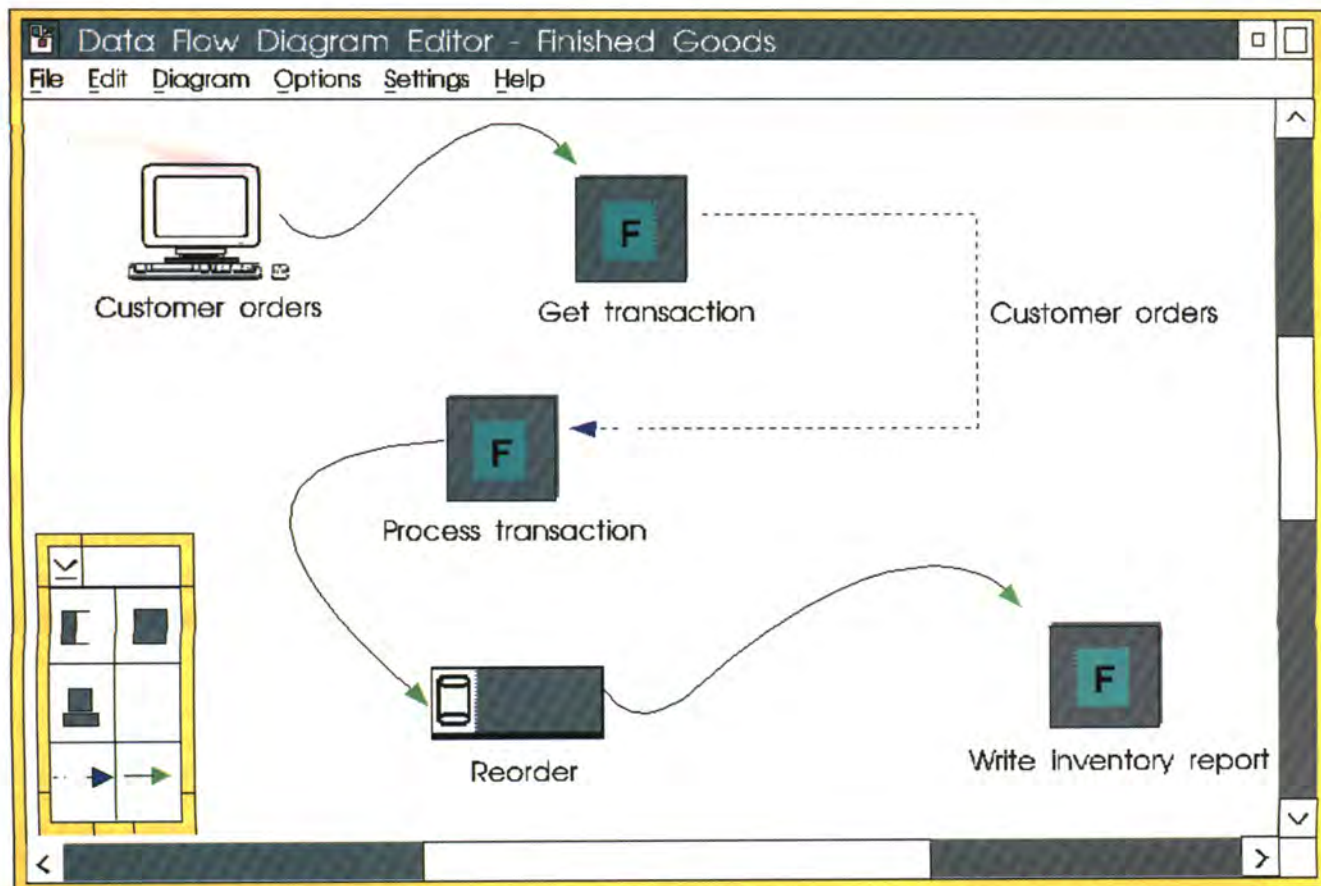


Figure 2. Data flow diagram editor built with Graphics Interface Kit/2

All manipulation of the symbols, such as selection, creation, and deletion, is done in the main window of the graphics editor. In addition, the graphics editor supports the following win-

- color of symbols)
- Clipboard view window (for displaying the contents of the clipboard).

*The graphics editor contains approximately 50 standard actions that users might need to build or manipulate a graphical interface.*

dows, all of which can be customized with the design tool:

- Palette window (for adding symbols)
- Overview window (for scrolling and zooming)
- Color window (for changing the

### GENERIC ACTIONS

The graphics editor contains approximately 50 standard actions that users might need to build or manipulate a graphical interface. Called generic actions because they can be modified to suit a specific interface, these actions are split into implicit actions, invoked

directly with the mouse, and explicit actions, invoked from the menu bar.

Implicit generic actions include moving symbols, symbol parts, and bend points; attaching links to symbols; sizing nodes; adding a bend point to a link; selecting symbols or symbol parts; and scrolling and zooming the main window from the overview window. Explicit generic actions include adding, deleting, copying, and changing the Z-order of symbols; opening, saving, and printing the graphics; and applying a layout algorithm.

### EVENTS

The graphics editor can detect the occurrence of nearly 50 events, such as the selection, addition, movement, or deletion of a symbol, the display of a pull-down menu, or the click of a mouse button.

An event is an opportunity to specify special processing. You can tell the



graphics editor to call an event handler (C/C++ function) whenever a particular event occurs. Any special processing is defined in the event handler.

### GRAPHICS LIBRARY FUNCTIONS

Graphics Interface Kit/2 offers a library of about 300 high-level functions that can be called to modify the default behavior of GIK/2 while an event is being processed, to connect graphical objects to application data, and to open and close GIK/2 windows from the application main code. For example, there are functions for:

- Initializing and quitting GIK/2 windows
- Inserting and removing choices in the menu bar
- Opening, zooming, printing, and saving the main window
- Linking application data to the main window
- Retrieving and displaying help panels
- Creating and deleting symbols
- Changing symbol sizes, text labels, and the position of symbols
- Setting a flag indicating that a symbol or symbol part should be affected by a later action
- Retrieving symbol information (position, size, overlap)
- Linking application data to symbols
- Selecting a layout control and selecting its parameters.

### CUSTOMIZATION WITH C-CODE

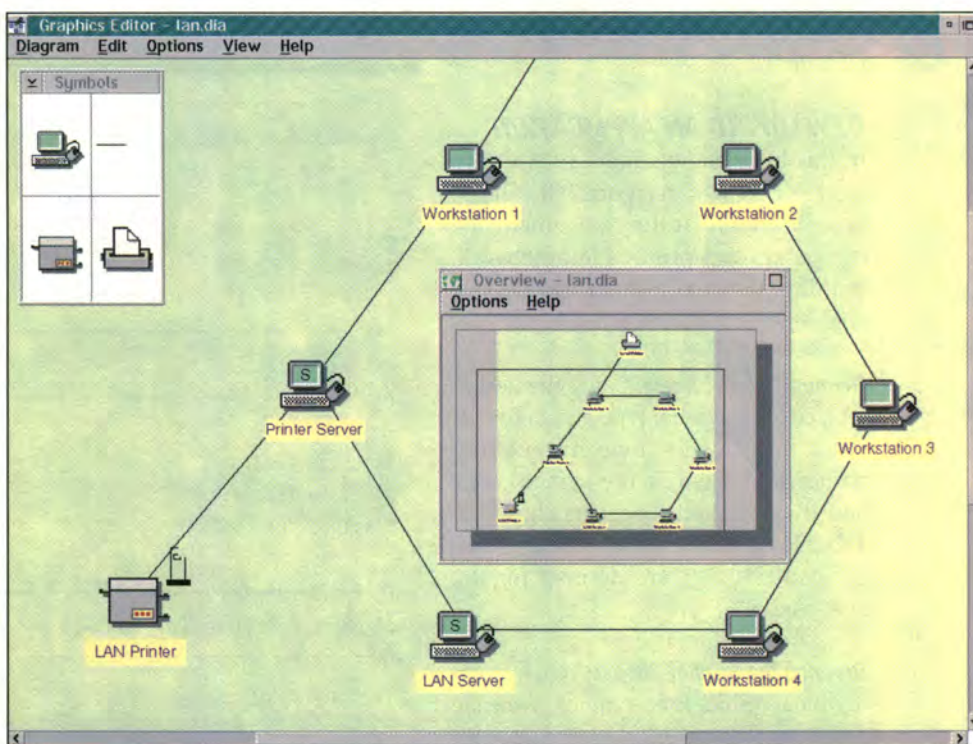
After developers have specified a design, they can add customizing code to build a running application. The customizing code will add the application logic and link the application data to the graphical interface.

### EVENT HANDLERS

Events and event handlers are the key link between the interface graphics and the application data. Event handlers must be registered in the design tool, which then generates code templates

|                        |                                                                                                                                                                                          |
|------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>LAN Layout</b>      | This layout arranges all nodes connected directly or indirectly to a central site. This algorithm arranges root nodes in a circle and the other nodes as trees growing from these nodes. |
| <b>Radial Layout</b>   | This layout arranges nodes as a set of recursive circles. The nodes are clustered in a circular shape.                                                                                   |
| <b>Horizontal Tree</b> | This layout arranges nodes as a horizontal tree. The roots are at the top and the leaves are at the bottom.                                                                              |
| <b>Vertical Tree</b>   | This layout arranges nodes as a vertical tree. The roots are at the left and the leaves are at the right.                                                                                |
| <b>Ellipse</b>         | This layout arranges nodes in a single ellipse. The algorithm repositions the nodes in the order of occurrence.                                                                          |

**Table 1.** Layout algorithms available with Graphics Interface Kit/2



**Figure 3.** Network topology drawn with the LAN Configurator example

for them. The collection of event and action handling functions, called customizing code, resides in an OS/2 dynamic link library.

Event handlers can be used for many purposes, such as:

- Map graphical editing actions to modifications of the underlying application data represented by the graphical objects. For example, when a user deletes a link in the diagram, a relationship in a server





- database may be deleted.
- Associate application-defined behavior with user actions. For example, when a user clicks with mouse button 2 on a specific symbol, a dialogue for data entry may be displayed.
- Modify the behavior of the graphics editor. For example, users could be prohibited from moving a specific symbol outside a defined area in the diagram.

### ACTION HANDLER

With the action handler, you can supply application-specific actions that users can select from the menu bar. For these application-specific functions, use the action handler, a C function called when a user selects a menu choice associated with it. It contains a large switch statement, with one case for each application-specific menu choice. Each choice has a unique identifier associated with it.

### DEVELOPING AN APPLICATION

In this example we build a LAN configurator, shown in Figure 3, that helps system administrators document the computers and printers in a network. Building the configurator can be broken down into six main tasks:

**Defining Symbol Types.** Using the design tool, create a symbol type and define its structure. This is done by entering names that identify the symbol type and its parts and forms, as shown in Table 1. No additional parts to the pre-defined LINK.BODY are defined for the LINK symbol.

**Drawing the Symbol Shapes.** When creating the graphics for a symbol, associate one or more graphical elements with each symbol part. A symbol part, which can be composed of bitmaps, icons, pointers, polylines, ellipses, boxes, and arcs, will contain different graphical elements for different forms of a symbol type, allowing you to build complex symbols. The symbol shapes are drawn with an interactive WYSIWYG editor.

|                          |                 |                       |
|--------------------------|-----------------|-----------------------|
| <b>Symbol types</b>      | PC              |                       |
|                          | PRINTER         |                       |
|                          | LINK            |                       |
| <hr/>                    |                 |                       |
| <b>Parts for PC</b>      | PC_SYMBOL       |                       |
|                          | NETWORK_NAME    | (textual symbol part) |
|                          | SERVER          | (initially invisible) |
| <b>Parts for PRINTER</b> | PRINTER_SYMBOL  |                       |
|                          | PRINTER_NAME    | (textual symbol part) |
|                          | NETWORK_PRINTER | (initially invisible) |
| <hr/>                    |                 |                       |
| <b>Forms for PC</b>      | PC_FORM         |                       |
| <b>Forms for PRINTER</b> | LASER_PRINTER   |                       |
|                          | MATRIX_PRINTER  |                       |
| <b>Forms for LINK</b>    | LINK_FORM       |                       |

Table 2. Symbol structure of the LAN Configurator example

```
/*-----*/
/* INCLUDE RELATED DEFINES */
/*-----*/

#define INCL_DOS /* OS/2 definitions */
#define INCL_PM /* PM definitions */

/*-----*/
/* HEADER FILES */
/*-----*/

#include <os2.h> /* OS/2 header file */
#include <ewyga.h> /* GIK/2 header file */
#include "LANCONF.H" /* Generated header file */

/*-----*/
/* LOCAL FUNCTION PROTOTYPES */
/*-----*/

SHORT GSETRY MyActionHandler(DHND,USHORT,MPARAM,MPARAM);

/*****
/* MyActionHandler */
/*
/* Parameters:
/* DHND dhnd (I): Diagram handle.
*****/
```

Figure 4. The action handler changing selected printer symbols to network printer symbols (Automatically generated code lines are printed in bold text.) (continued on page 65)



**Defining a Symbol Palette.** In the design tool, specify the number of rows and columns in the palette, then associate a symbol form with each cell of the palette.

**Testing the Design.** At this point, the test action can be used for the first time. You get a full-function graphics editor in which graphical symbols can be added, deleted, or manipulated, but there is no connection to the application data.

**Specifying the Layout Control.** The name of the layout control can be one of the GIK/2-provided controls or a self-written control. Menu choices added to the menu bar enable users to lay out the diagram. Table 2 shows layout algorithms available with GIK/2.

**Writing Customizing Code.** The C-code used to customize the default behavior of the generic editor must be contained in a DLL whose name is specified in the design tool.

You can add special processing to the generic editor by defining either an action handler or an event handler.

An action handler is invoked when a user selects a choice from the menu bar. In this example, a menu choice turns all selected printer symbols into network-printer symbols. To achieve this, add a menu choice to the default menu bar, register the C function `MyActionHandler` as an action handler, generate the skeleton code, and add the specific code to the skeleton, as shown in Figure 4.

An event handler is invoked when GIK/2 detects one of the predefined events. In this example, the graphics editor is customized so that a requester symbol is changed to a server symbol when a user double-clicks on it. To achieve this, register the C function `MyDoubleClick` to the event `EV_DOUBLE_CLICK`, generate the skeleton code, and add the specific code to the skeleton, as shown in Figure 5.

The following files can be generated by the design tool to support the development of customizing code:

```
/* USHORT menu_id (I): Action selected by user. */
/* MPARAM mparam1 (I): PM-data. */
/* MPARAM mparam2 (I): PM-data. */
/* Returns: */
/* GS_OK */
/* Description: */
/* The function handles all non-generic actions. */
/*****

SHORT GSEENTRY MyActionHandler(DHND dhnd,
                                USHORT menu_id,
                                MPARAM mparam1,
                                MPARAM mparam2)
{
    SHND shnd; /* symbol handle */
    SHORT sym_type; /* symbol type */
}
```

**Figure 4.** The action handler changing selected printer symbols to network printer symbols. (Automatically generated code lines are printed in bold text.) (continued on page 66)

## Looking for fast answers to your development questions?

**You need to do two things:  
Go on-line with CompuServe  
Information Service. Use Golden  
CommPass to do it.**

CompuServe hosts OS/2 developer and user forums monitored by technical personnel. IBM developers are there with responses to anything that's got you stuck. Other OS/2 programmers and enthusiasts are there, too, comparing notes and giving feedback. It's definitely the place to be.

**Time is money when you connect to CompuServe, and Golden CommPass saves you both.** It lets you compose your questions off-line, send them in a flash and disconnect. It gets your replies while you're busy programming. It's a native OS/2 app, with all the power and features you'd expect.

**Get the fastest access to answers.** Golden CommPass keeps your development schedule on course. The fact is, the sooner you start using our program, the sooner they'll be talking about yours!



**CompuServe:**  
71511, 151  
**90 Day Satisfaction Guarantee**

Phone:  
(609) 234-1500  
Fax:  
(609) 234-1920



**Golden CommPass™**  
CompuServe® Access Software

Creative Systems Programming Corporation



## REFERENCES

Graphics Interface Kit/2 (IBM Doc. 5621-432) can be ordered from IBM directly or from authorized personal computer dealers and software distributors. You can also order this program and request additional information or a demonstration package by calling 1-800-426-2255 and asking for Department S71. In Europe, send a fax to 43-1-21145-4490, Attn: GIK/2.

The program is also available in French, German, Italian, and Japanese. The following IBM documents are also useful:

*Graphics Interface Kit/2 Getting Started* (IBM Doc. SH19-8189)

*Build Intelligent Graphics With Graphics Interface Kit/2* (IBM Doc. G511-1785)

- Include file—Defines the constants for the symbol type and other design characteristics.
- Action handler—C file containing the skeleton code for handling application-defined menu choices.
- Event handler—C file containing the function skeletons for all registered events.
- Definition file—Defines the customizing code DLL.
- Make file—Creates the customizing code DLL.

```
switch (menu_id)
{
    case AID_NETWORK_PRINTER : /* Network Printer */
        GsCollectSym(dhnd, GS_NODE, GS_SYM_SELECTED, GS_LOWER_FIRST);
        while(GsGetCollectedSym(dhnd, &shnd) == GS_OK)
        {x
            GsGetSymType(dhnd, shnd, &sym_type);
            if (sym_type == PRINTER)
            {
                GsPutPartVisibility(dhnd, shnd, NETWORK_PRINTER, GS_ON);
            }
        }
        break;
    }
    return GS_OK;
}
```

**Figure 4.** The action handler changing selected printer symbols to network printer symbols (Automatically generated code lines are printed in bold text) (continued from page 65)

```
/*-----*/
/* INCLUDE RELATED DEFINES */
/*-----*/

#define INCL_DOS /* OS/2 definitions */
#define INCL_PM /* PM definitions */

/*-----*/
/* HEADER FILES */
/*-----*/

#include <os2.h> /* OS/2 header file */
#include <ewyga.h> /* GIK/2 header file */
#include "LANCONF.H" /* Generated header file */

/*-----*/
/* LOCAL FUNCTION PROTOTYPES */
/*-----*/

SHORT GSENTRY MyDoubleClick(DHND,SHORT,EVS_DOUBLE_CLICK*);

/*****
/* MyDoubleClick */
/*
/* Parameters: */
*****/
```

**Figure 5.** The event handler changing a requester symbol to a server symbol when a double-click occurs (Automatically generated code lines are printed in bold text) (continued on page 68)



Install With Ease, Install With Style,  
Install With The Best

## Software Installer

for OS/2 2.0 & 2.1

Why spend your valuable time developing the installation processes for your applications when Software Installer makes the job much easier? You can use its extensive services to install, update, restore, and delete your products. End users can install your applications from diskette, CD ROM, Local Area Networks (LAN), or from MVS, OS/400, VM or VSE host systems.

Software Installer provides routines for:

- Creating animated Presentation Manager™ installations
- Updating the CONFIG.SYS file
- Adding, deleting, and changing OS/2 files, INI files and Workplace Shell™ classes and objects

Many other features are included, such as:

- Multi-threaded processing
- Disk generation utility
- Open and programmable interfaces
- Response file invocation for non-interactive installations

For information on Software Installer for Windows™ call 1-800-IBM-CARY

**List \$349**

for more information call:

**Call 1-800-IBM-CARY**

to order call either:

**Indelible Blue 1-800-776-8284**  
**Business Depot 1-800-844-8448**

Let Distributed Application/2  
Do The Talking!

## Distributed Application/2™

Distributed Application/2 is a set of application programming interfaces (APIs) that provide a consistent way to access interprocess and network communication functions under OS/2 2.0, making client/server applications easier to create than ever. Distributed Application/2 and its APIs:

- Will help you compete in the growing client/server programming arena
- Provide an API set for interprocess and network communications
- Hide the complexity of the supported protocols
- Enable you to select a protocol at run time, without changing your code
- Allow you to focus on your application, and leave the client/server communication to Distributed Application/2

Many other features are included, such as:

- Multiple protocols: APPC, NetBIOS (IBM and Novell®), OS/2 named pipes
- Access to IMS transactions
- C and REXX language interfaces (+ any other language using the C calling conventions)
- Full duplex communications
- Easy-to-use connection definition tool

For information on Distributed Application/2™ call 1-800-IBM-CARY

**List \$299**

## Software Installer



Developed by  
IBM Programming Systems

**Now Creates  
CID Enabled  
Install Programs!**

## Distributed Application/2™



Developed by  
IBM Programming Systems

**Unleash  
Client/Server  
Programming!**





**Silvia Hansjakob**, IBM Programming Systems Line of Business, Lassallestrasse 1, A-1020 Vienna, Austria. Hansjakob is a development manager in the IBM Vienna Software Development Laboratory. She joined IBM in 1988 and has worked in graphical user interface development since then. She holds a doctorate from the University of Vienna.

**Roland Hatschka**, IBM Programming Systems Line of Business, Lassallestrasse 1, A-1020 Vienna, Austria. Hatschka is a program manager in the Vienna Software Development Laboratory. He joined IBM in 1988 and has worked in graphical user interface development since then. He holds a master's degree and a doctorate in computer science from the Technical University of Vienna.

```

/* DHND          dhnd (I): Diagram handle.          */
/* SHORT         ev_id (I): Identifies the event.    */
/* EVS_DOUBLE_CLICK *pEvs (I): Pointer to instance of event*/
/*                                     data type.      */
/* SHND  shnd (I):      Handle of the symbol          */
/* SHORT part_ref (I):  Reference number of the symbol part*/
/* USHORT button (I):   Mouse button that is pressed   */
/*                                     It is one of the following: */
/*                                     GS_MB1             */
/*                                     GS_MB2             */
/*                                     */
/* Returns:                                             */
/* GS_NO      Any changes made to symbols remain and the action */
/*            stops.                                           */
/* GS_NO_1    Any changes made to symbols remain and the action */
/*            stops.                                           */
/* GS_RESTORE Any changes made to symbols since the last undo */
/*            checkpoint are undone and the action stops.      */
/* GS_RESTORE_1 Any changes made to symbols since the last undo */
/*            checkpoint are undone and the action stops.      */
/* GS_YES     The default processing continues.               */
/*            */
/* Description:                                           */
/* The function handles the event EV_DOUBLE_CLICK.        */
/*****

SHORT GSENTRY MyDoubleClick(DHND dhnd,
                           SHORT ev_id,
                           EVS_DOUBLE_CLICK *pEvs)
{
    SHORT sym_type;          /* symbol type */
    SHORT visibility;        /* visibility of symbol part */

    if (pEvs->button == GS_MB2)
    {
        GsGetSymType(dhnd, pEvs->shnd, &sym_type);
        if (sym_type == PC)
        {
            /* toggle visibility state of part, showing server */
            GsGetPartVisibility(dhnd, pEvs->shnd, SERVER, &visibility);
            GsPutPartVisibility(dhnd, pEvs->shnd, SERVER, !visibility);
        }
    }
    return GS_YES;
}

```

**Figure 5.** The event handler changing a requester symbol to a server symbol when a double-click occurs (Automatically generated code lines are printed in bold text) (continued from page 66)



# Fax at a higher level

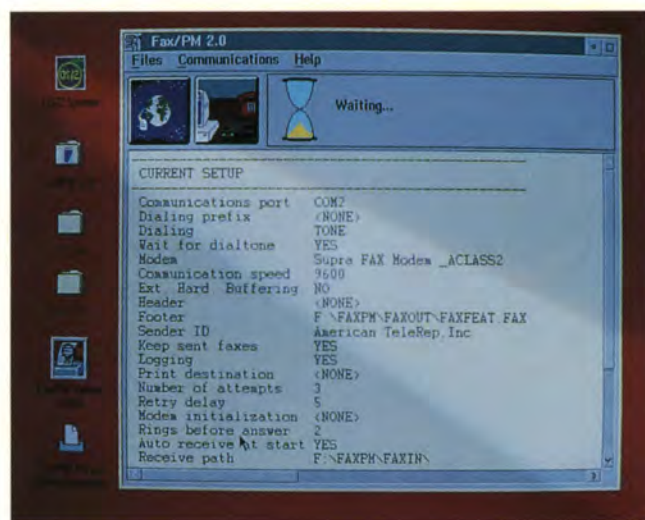


Whether you're in a Dos, Win-OS2 or OS/2 session running under OS/2 2.1, now you have the ability to send and receive faxes - without leaving the application. Just select the Fax/PM driver as your printer. To fax...just print!

Fax/PM 32-bit also really capitalizes on the advanced features of the Workplace Shell object-oriented interface. You can send faxes right from the Workplace Shell by simply dragging and dropping a data object to the fax object.

Features like these are too good to keep to yourself, so there are also Fax/PM versions available for LAN and Client-Server environments, all CID enabled.

And applications developers will appreciate SOM APIs that allow for integration of the Fax/PM engine into custom applications. We could go on about how Fax/PM can make your PC an even



more productive place. But we'd rather let the fax speak for themselves.

# with Fax/PM.

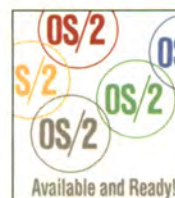
To order or to find out more about Fax/PM, call U.S.: 1-203-644-1708 / fax 1-203-648-9587  
Europe: (33) 1 48 70 19 00 / fax (33) 1 48 70 27 29



## Microformatic

Europe  
2 rue Navoiseau  
F-93100 Montreuil  
France.

U.S.A.  
P.O. Box 722  
610 Niederwerfer Road  
South Windsor, CT 06074







**GUI**

*OS/2 Presentation Manager provides an environment to develop object-oriented and device-independent applications with consistent graphical user interfaces. This article describes the major features, components, and inner workings of Presentation Manager.* **BY PYLEE LENNIL**

# Demystifying the OS/2 Presentation Manager

**P**resentation Manager (PM), the primary application environment under OS/2, deals with menus, dialogue boxes, and scroll bars accessible through either the keyboard or mouse. It allows applications to share a video display, mouse, and printer with other applications in a fixed windowing environment. PM provides three major features: an object-oriented programming environment, consistent user interface, and device independence.

Presentation Manager provides an object-oriented, message-based programming environment. In the object-oriented approach, an object consists of data and a set of rules that describe how to

One of PM's goals is to provide capabilities for creating applications that conform to the Common User Access (CUA) guidelines of the IBM Systems Application Architecture (SAA). The goal of SAA is to provide a consistent, easy-to-use user interface across IBM system applications. Conforming to SAA guidelines significantly reduces the learning curve for new applications and also minimizes the effort required to port the applications to other CUA systems. Presentation Manager CUA support provides capabilities such as menus, dialogue boxes, and scroll bars that can be used to generate applications that conform to CUA.

PM's environment makes applications device-independent; applications need not be aware of the characteristics of a specific device such as a display, printer, or plotter. Applications that access devices using the PM API automatically become device-independent and will run on many devices, provided that PM supports a driver for them.

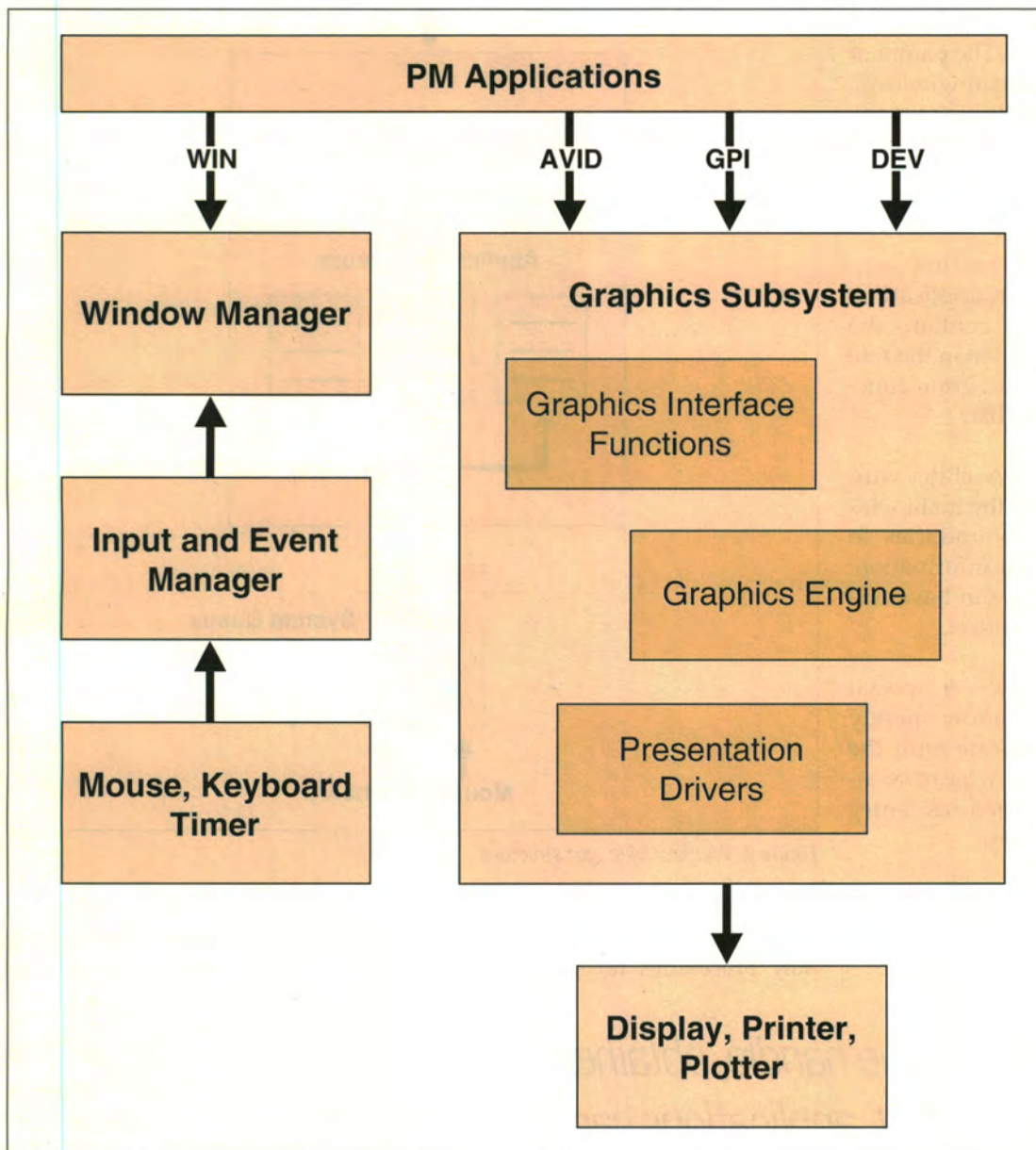
## **STRUCTURE OF PRESENTATION MANAGER**

PM consists of a window manager, graphics subsystem, and input and event manager, shown in Figure 1. These form an extension of OS/2 that provides window management, graphics support, and input and event support. Application window management is provided by the window manager and graphics support by the graphics subsystem. The input and event manager manages input events from the mouse, keyboard, and system timer. The following section examines these components in detail.

*PM applications are object-oriented, message-based, CUA-conforming, and device-independent.*

manipulate the data. The Presentation Manager implementation of the object consists of a window and a procedure. The window is manipulated by invoking the procedure by sending a message to the window. Messages can originate from the user (mouse or keyboard), from the PM timer, or from other PM applications. For instance, a window can be closed or minimized by a message event generated by a mouse click.





**Figure 1.** Presentation Manager structure

**Window Manager.** The window manager, which manages application windows and message queues, contains a set of window functions and application message queues, as shown in Figure 2. An application issues PM window function (WIN) calls to invoke these functions. A window receives input in the form of messages and displays output to a device through presentation space. Messages, which can originate from the mouse, key-

board, system timer, or other applications, are used to manipulate windows.

An application creates a window with the `WinCreateWindow` function. Applications can create many windows, each used for a dialogue between the application and user. A window can be moved, minimized, or maximized with input from the user. Data in a window can also be moved or copied from one window to another or between

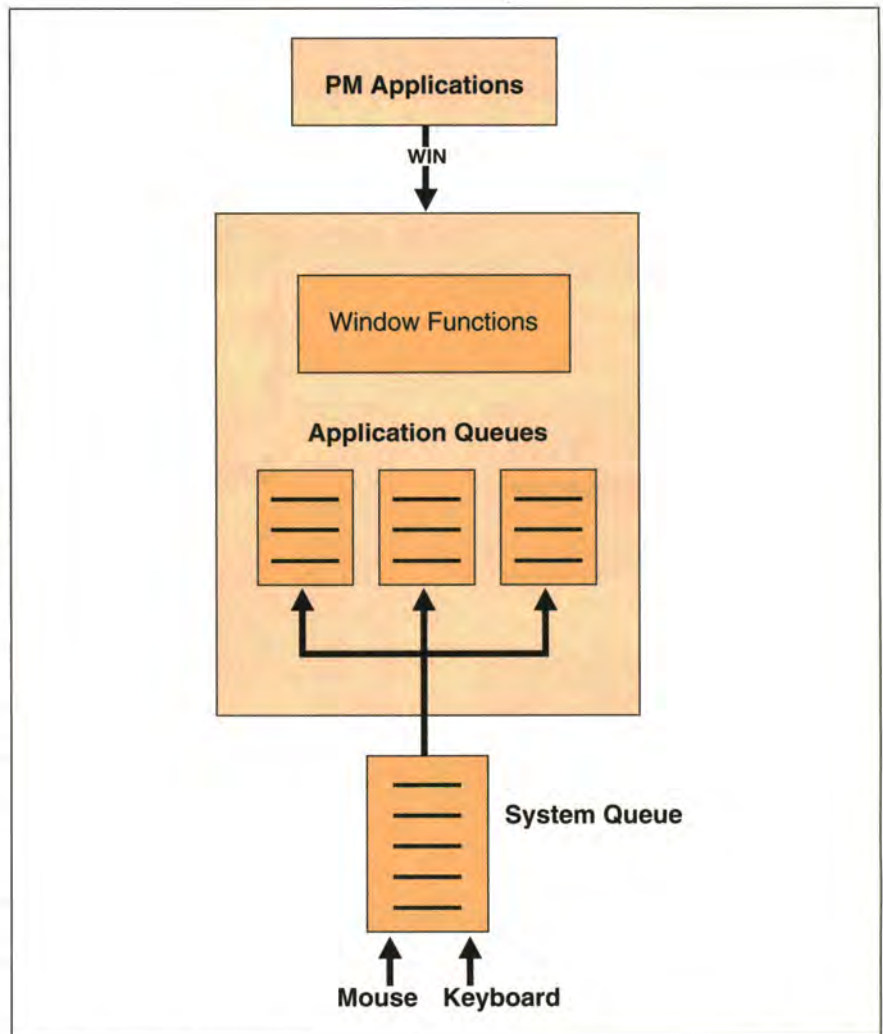


areas of the same window using the Presentation Manager clipboard functions (cut, copy, and paste). Presentation Manager provides several types of windows to support CUA guidelines:

- Desktop window—The parent of all applications' main windows, the desktop window shows the entire screen when it is used by PM in a windowing environment.
- Main Window—The first window created by an application, the main window contains the title of the application in the title bar and a list of program functions in the action bar.
- Child Window—A child window is created by the main window within its boundaries to display additional information. A parent window can have one or more child windows.
- Dialogue Window—A special type of child window mainly used to communicate with the user, the dialogue window contains controls such as entry fields and list boxes.
- Message Window—The message window displays messages to the user.

*Each window has a unique handle, obtained when it is created. With it, applications can send messages to a destination window using the **WinSendMessage** or **WinPostMessage** functions.*

The window with which a user is interacting, or the active window, receives keyboard or mouse input. Each window has a class that associates it with a window procedure, which defines the behavior of windows in the



**Figure 2.** Window Manager structure

class. There are two types of window classes: public and private. The window procedures for public classes,

class during application initialization; this associates the class with a window procedure. The application can then invoke the window procedure by dispatching a message.

Presentation Manager's object-oriented programming environment is implemented in the window manager. The window is the object; a window procedure is associated with a window class. The window is manipulated by sending messages to invoke specific functions in the window procedure that change window behavior. Each window has a unique handle, obtained when it is created. With the handle, applications can send messages to a destination window using the **WinSendMessage** or **WinPostMessage** functions. Messages from the mouse, keyboard, and system timer are stored in system



```

main(      )      /* main procedure      */
{
    WinInitialize(); /* initialize application */

    WinCreateMsgQueue(); /* create application queue */

    WinRegisterClass(); /* register window procedure */
                        /* to the window class */

    WinCreateStdWindow() /* create main window */

                        /* message processing loop */
    while( WinGetMsg()); /* get message from queues */

        WinDispatchMsg(); /* dispatch message to */
                        /* window proc */
}

```

Figure 3. Application main procedure

queue. PM applications can retrieve these messages with `WinGetMsg` and send to the window procedure with `WinDispatchMsg`.

**Messages.** Messaging is the primary way that PM manipulates windows. Messages originate from the mouse, keyboard, PM timer, application, or operating system. A mouse message is generated by mouse movement or by clicking the mouse button; a keyboard message is generated for every key stroke. A timer message is generated when a PM timer has elapsed. Application messages are sent by an application to other application windows. A mouse or keyboard message has the following structure:



## The Path To A Good GUI Isn't Always Clear

### Navigate the maze with the NEW 2.1 Professional Developers ToolKit

Promises are cheap, GUI development tools aren't. Gpf 2.1 demo software is **FREE**. Try Gpf before you buy any tool and you'll agree that **Gpf 2.1 DELIVERS**.

With this powerful point and click visual programming environment you can create a CUA '91 Graphical User Interface for OS/2 Presentation Manager® or MS DOS/Windows® in as little as 10% of the time required to hand code the same design.

What You See Is What You Get programming reduces weeks of coding to hours. Quickly prototype and test your interface, then let **Gpf write the code**. Gpf generates error free, structured Object Oriented C or C++ source, complete with embedded SQL for OS/2 DataBase. Custom Help and Logic are added to controls as they are drawn to create full **Client/Server** or **Stand-alone** applications! Gpf is hosted on OS/2 2.x and generates **native code** for OS/2 2.x, OS/2 1.3.x and MS Windows 3.0 or 3.1 Of course this means maximum performance with **no run time module** and **no royalties!**

#### Seeing Gpf is believing!

- Design 32 or 16 bit GUIs for IBM OS/2® or Microsoft Windows®
- Be creative, **WYSIWYG** design environment
- Full application and/or **DLL** generation
- Minimal time to application delivery
- Full CUA '91 Control set, 32 or 16 bit
- **Reusable Objects**
- Automatic documentation

# Gpf

Try us, **FREE** Demo Software Available  
 Order Gpf Pro ToolKit today for just \$1440<sup>00</sup>  
 Separately: Gpf 2.1 for \$1295<sup>00</sup> & GpfTools for \$195<sup>00</sup>  
 Or, try: Gpf 2.0 Single Platform, now available for just \$495<sup>00</sup>

Call Gpf Systems Inc. at: (800) 831-0017



\* GUI Programming Facility



**SYSTEMS, INC.** Phone (203) 873-3300 • fax (203) 873-3302 30 Falls Road, Moodus, CT 06469



- Window handle
- Message ID
- Parameter 1
- Parameter 2
- Time
- Pointer data.

The window handle identifies the destination application window, while the message ID identifies the message type. For instance, an ID of 7a hex identifies a keyboard message (WM\_CHAR) and 71 hex identifies that mouse button 1 is down (WMBUTTONDOWN). Parameters 1 and 2 contain information depending on the message type. For instance, a keyboard message Parameter 1 and 2 in a keyboard message contain scan code, control code, and other information from the pressed key. Time fields contain the time that a message was sent and the pointer data contains the position of the mouse pointer when the message was sent.

**Message Queues.** PM maintains two types of message queues, a system queue and an application queue. (See Figure 2.) The system queue, created during PM's initialization time, is used to store mouse and keyboard messages. Messages are removed from the system queue in a first in, first out order, ensuring the same sequence as the user input. Timer messages are not stored in the system queue because they are processed immediately to update the timers set by the applications.

An application queue, created for every application and its associated application threads with `WinCreateMsgQueue`, receives messages from other applications. Messages are sent between applications with `WinSendMessage` or `WinPostMessage`, retrieved by `WinGetMsg`, and dispatched to the window procedure with `WinDispatchMsg`.

The application that processes the messages consists of a main procedure and a window procedure. The main procedure, shown in Figure 3, performs several tasks:

- Initializes PM facilities for the application with `WinInitialize`

```
WindowProc( )           /* window procedure */
{
    switch( msg )
    {
        /* if character message */
        case WM_CHAR:
            [ Process character ]

        /* if mouse move message */
        case WM_MOUSEMOVE:
            [ Process mouse movement message ]

        /* if destroy window message */
        case WM_DESTROY:
            [ Do clean up ]
    }

    return WinDefWindowProc( ); /* default window procedure */
}
```

Figure 4. Application window procedure

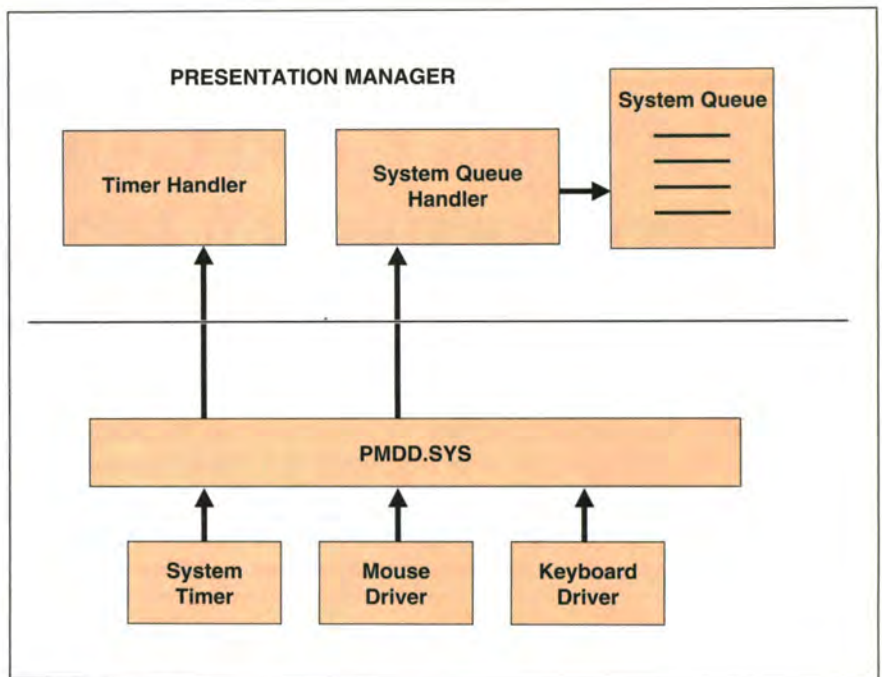


Figure 5. Input and event manager structure

- Creates an application queue with `WinCreateMsgQueue`
- Registers the window procedure in Figure 4 to the window class with `WinRegisterWindowClass`
- Creates an application main window using `WinCreateStdWindow`
- Performs message looping using

`WinGetMsg` and `WinDispatchMsg` to read messages from queues and dispatch them to the window procedure shown in Figure 4.

Message processing is initiated when the application issues a `WinGetMsg` function, as shown in Figure 3. The function scans the system queue for messages specific to the application. If a message is found, it returns it to the



application's main procedure. The main procedure, in turn, dispatches the message to the window procedure with `WinDispatchMsg`. Next, `WinGetMsg` scans the application queue for any messages specific to the application. When one is found, it retrieves and dispatches it to the window procedure with `WinDispatchMsg`.

The window procedure consists mainly of a multiple selection switch statement with a case for each message, as shown in Figure 4. The switch statement selects appropriate routines to manipulate the window. All nonrelevant messages are handled by a default window procedure, `WinDefWindowProc`, which ignores the message. The control is then returned to the main procedure, which continues scanning for more messages in the message queues.

**Input and Event Manager.** The input and event manager processes input events from the mouse, keyboard, and timer. As shown in Figure 5, input events are routed to PM through the kernel device driver `PMDD.SYS`. The mouse and keyboard events are processed by the system queue handler, while the timer tick is processed by the timer handler. The mouse event data consists of information such as coordinates of the current mouse position; the keyboard event consists of scan code and control code. The system queue handler converts the mouse and keyboard event data into message packets and stores them in the system queue. A timer event is generated for every timer tick and the timer handler updates the timer data structure set up by the PM applications. If any timer has elapsed, it sends the message `WM_TIMER` to the appropriate application queue. Applications issue a `WinGetMsg` function to retrieve messages from the system queue or a `WinPeekMsg` to examine the messages. The messages are then dispatched to the window procedure with a `WinDispatchMsg` function call.

**Graphics Subsystem.** The graphics subsystem allows the application to create, display, and retrieve graphics, text, and

pictures in a device-independent manner. The subsystem consists of a graphics program interface (GPI), graphics engine (GRE), and presentation drivers, illustrated in Figure 6.

The GPI programming interface consists of a set of GPI functions that create, manipulate, and display graphics objects such as lines, arcs, and character fonts. GPI functions call the graphics engine to manage output.

## PRESENTATION SPACE

An application must create a presentation space to manage graphics it creates before it can draw on a device. The presentation space is a data area maintained by the presentation manager. The application writes graphics attributes such as color or line style and transforms them to the presentation space. These device-independent graphics attributes are translated to device-dependent data by the device context before being sent to a device. The application must therefore associate the presentation space with a context-specific device before data can be written to that device.

There are two types of presentation spaces: normal and micro. The type of presentation space chosen by an application is determined by the type of drawing.

**Normal Presentation Space.** Normal presentation space is used by applications that use graphics segments or retained-drawing functions to generate complex pictures. A graphic segment is a collec-

during program initialization and destroyed during program termination time, normal presentation space requires a large amount of memory and is intended to be used by an application for long periods of time. Normal presentation space also has slower performance.

**Micro Presentation Space.** Micro presentation space is normally used to generate simple drawings. There are two

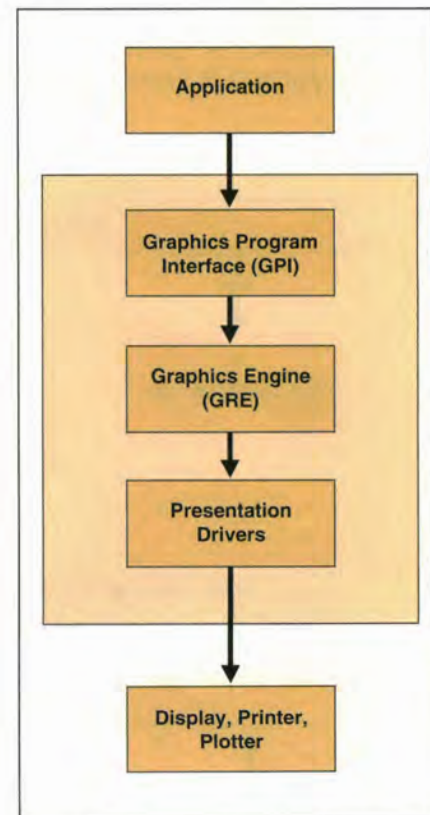


Figure 6. Graphics subsystem structure

*PM's graphics subsystem lets your application create, display, and retrieve text and graphics.*

tion of graphics, drawings, and attribute commands stored in the presentation space. The major advantage of using normal presentation space is that the application can use the space to write to devices such as video displays, printers, and plotters. Created

types of micro presentation space: normal and cached.

Although normal micro presentation space can be used to write to multiple devices, the application must use a separate presentation for each device. It requires less memory space and pro-



vides better performance. Normal presentation space is used for short periods of time and destroyed after performing a drawing task. Its major disadvantage is that it does not support graphics segmentation and GPI segment functions. Normal micro presentation space is created with `GpiCreatePS` and destroyed with `GpiDestroyPS`.

Cached micro presentation space is used only by the window manager to display a window on a video display. Always associated with a video display, it cannot be used to draw on other devices such as a printer or plotter. The application opens a cached

micro presentation space using `WinOpenWindowDC`.

### DEVICE CONTEXT

Device context describes the characteristics of display devices such as the video display, printer, and plotter. The device context structure describes a physical device and conveys this information to the presentation space associated with it. It then translates device-independent graphics to device-dependent drawings. A window device context is opened with `WinOpenWindowDC` and a nonwindow device context with `DevOpenDC`. Each open call creates an

instance of a device context, which is destroyed when the application closes it with `DevCloseDC`. The steps needed to use presentation space and the device context to write graphics data on a device are:

1. Open device context
2. Create presentation space
3. Associate presentation space with device context
4. Draw text and graphics to presentation space
5. Destroy presentation space.

### GRAPHICS ENGINE

The primary functions of the graphics engine are to provide graphic simulation and manage graphics output to display drivers. These functions, which can be accessed with the PM GRE function calls, are available to display drivers through a function dispatch table maintained by the graphics engine. (See Figure 7.) The table, which contains entry points to the functions, is shared by the graphics engine and display drivers. The graphics engine passes a copy of the dispatch table to each presentation driver when it is loaded. The presentation driver, in turn, fills the table with its own function entry points.

### PRESENTATION DRIVERS

The presentation driver is a higher-level interface to display, printer, and plotter devices. Unlike kernel device drivers that reside in ring 0, the presentation driver resides in ring 2 or ring 3. It provides graphics functions for display and printer devices; these functions are accessed through a presentation driver interface (PDI). Presentation drivers' I/O privileges allow them to access the devices directly. Complex graphics functions that cannot be implemented by the presentation drivers are implemented by the graphics engine. These functions can be directly called from the presentation driver using a dispatch table mechanism, as shown in Figure 7. When a device context is initialized, the dispatch table is passed to the presentation driver by the

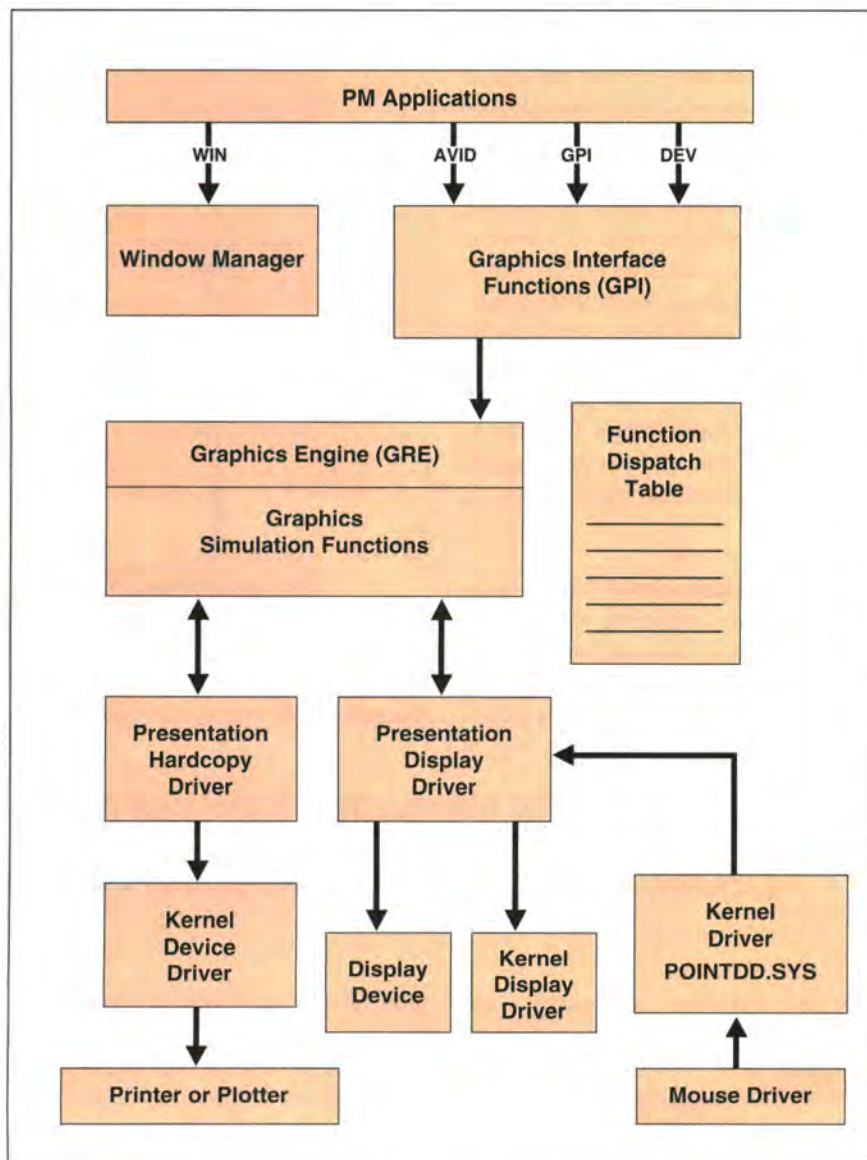


Figure 7. Graphics engine and presentation driver



graphics engine. There are two types of presentation drivers, the display driver and the hardcopy driver.

**Presentation Display Driver.** The main function of the presentation display driver is to display graphics data on video display devices. The driver must execute quickly to update the display screen, writing directly to the video screen using the I/O privilege. The display driver can call kernel display device drivers to obtain display adapter hardware and video memory buffer addresses. It calls the graphics engine for complex graphics functions not implemented in the display driver. The presentation display driver is loaded and initialized when the presentation manager is initialized. The display driver is also used by the mouse driver through POINTDD.SYS to draw the mouse pointer on the screen.

**Presentation Hardcopy Driver.** The hardcopy driver draws graphics on printers and plotters. It supports different forms, provides a dialogue for user configuration, and calls kernel drivers to simulate certain functions. Printer drivers call the graphics engine to support complex graphics functions. Unlike the display driver, the hardcopy driver is loaded not during PM initialization but when the application issues a DevOpenDC command to open the printer device context.

## HOOKS

Presentation Manager provides a hook mechanism that allows applications to monitor and modify the message stream. Hooks can be installed in the system or application queue; installing them into the system queue affects all applications, while the individual application queue affects only the queue's owner. Hooks are installed with WinSetHook and released with WinReleaseHook. The name of the hook procedure and the type of hook is specified with the WinSetHook function. Hooks can be categorized into six types:

the system queue. Used to monitor event data from the mouse and keyboard, it is called with WinGetMsg or WinPeekMsg.

2. Send-Message—The send-message hook monitors messages not posted to a queue. Using input and

send-message hooks, an application can monitor all window messages.

3. Filter—The filter hook provides input message filtering during the application's main procedure message loop. This allows applications to examine or modify the message



## SINGLE KEY-STROKE SCREEN PRINTING



CLIPBOARD  
VIEWER

SCREEN  
CAPTURE

- Create **Quality Black & White** copies of color images.
- Screen Saver prevents monitor burn-in.
- Date & Time Display for Time-Stamping images.
- LAN compatible.
- Entity & OEM licensing available.
- Economical combination of important utilities.
- The most stable and reliable product available.
- Includes both OS/2 2 (32-bit) and OS/2 1.3 (16-bit) versions.

THE LEADER: PrntScrn™ Screen Image Utility for OS/2



**MITNOR Software**  
Phone: (918) 357-1628  
Fax: (918) 357-2869

1. Input—The input hook monitors





before dispatching it to the window procedure.

4. **Journal-Record**—This hook monitors the system queue and allows the application to record input events. For example, it can be used to record a set of mouse and keyboard events and played back later using the journal-playback hook.

used but less flexible. The user invokes the clipboard using the cut, copy, or paste commands to transfer data within an application or from one application to the another. The only restriction on the clipboard is that data must be in a format that can be understood by the destination application. The application accesses the clipboard through a set of WIN functions; for example, an application issues a `WinOpenClip` function to

application. This mechanism can be used for activating a menu item by typing the keystroke. This is done by filtering the keyboard input and translating the keystroke to a corresponding command before reaching the application. For instance, to generate help messages, you can set up an accelerator (Alt-F8) to generate the `WM_HELP` message that invokes the application window procedure. The accelerators are kept in an accelerator table. Accelerators can be associated with the system queue or there can be one for each application window.

The system queue accelerator table affects all applications, while the individual window accelerator table affects only that application window. An application can modify both its own accelerator table and the system accelerator table. An application creates an accelerator table using the resource-definition file, which contains accelerator definitions that interact with a standard window when the window is created.

## *Presentation Manager provides two mechanisms to exchange data between applications, the clipboard and dynamic data exchange.*

5. **Journal-Playback**—This hook is used to play back a set of mouse and keyboard events previously recorded with the journal-record hook.
6. **Help**—The help hook is used while processing the `WM_HELP` message.

### **TIMER SUPPORT**

The PM Timer facility allows an application to set a timer with a specific duration. The application starts the timer with `WinStartTimer`. PM maintains a linked list of timer entries updated at each timer tick by the input and event manager. If any of the timer entries is empty, it sends a `WM_TIMER` message to the corresponding application queue. The application retrieves the message during the next `WinGetMsg` function and dispatches it to the window procedure, which in turn invokes a timer handler. The application can stop the timer with `WinStopTimer`.

### **DATA EXCHANGE**

Presentation Manager provides two mechanisms to exchange data between applications, the clipboard and dynamic data exchange (DDE).

The clipboard is more commonly

open the clipboard.

DDE is a data exchange facility that allows PM applications to exchange data using shared memory. Unlike the clipboard, DDE provides a one-time data exchange with user intervention, allowing continuous data exchange between server and client applications without any user intervention. Normally, the client initiates the data exchange by requesting data from the server, which responds by providing data to the client. An application can reside on both client and server and a client can request data from multiple servers; for example, a client application can request data from a server application and then act as a server by forwarding the data to another client.

Presentation Manager provides three functions to support DDE. A client initiates the data exchange using the `WinDdeInitiate` function. The server responds to the request using `WinDdeRespond`. The `WinDdePostMsg` function allows an application to post a message to another application with which it is carrying out a DDE.

### **ACCELERATOR TABLE**

An accelerator is a keystroke that generates a command message for an

### **ATOM TABLES**

Presentation Manager provides a mechanism for converting a character string into a 16-bit unique word. Each word, called an atom, has a unique ID and an associated counter, used to keep track of atom use. An atom table contains a collection of atoms and is maintained by the atom manager. When the counter becomes zero, the atom is deleted from the table. Using an atom instead of a string has many advantages. A string used in many data structures can be replaced by its atom and saves memory space. Searching an atom takes less time than searching a string. Atoms can be used for creating system-wide unique strings for window messages or DDE formats used between applications.

The atom table is maintained by a set of PM functions. `WinCreateAtomTable` creates an atom table, `WinAddAtom` adds an atom to the table, `WinFindAtom` finds an atom, and `WinDeleteAtom` deletes an atom from the table.



**Pylee Lennil**, IBM Corp., 1000 N.W. 51st St., Boca Raton, Fla. 33429, (407) 443-3855. Lennil joined IBM in 1983 and is an advisory programmer in the IBM entry systems technology laboratory. He is presently involved in the development of OS/2 and spent several years in PC DOS and AIX development. He received a B.S. in physics from Kerala University in India and an M.S. in computer engineering from the University of Massachusetts at Lowell.

## REFERENCES

Cheatham, Paul W., David E. Reich, and Robert F.G. Robinson. *OS/2 Presentation Manager Programming*. New York, N.Y.: John Wiley and Sons, 1990.

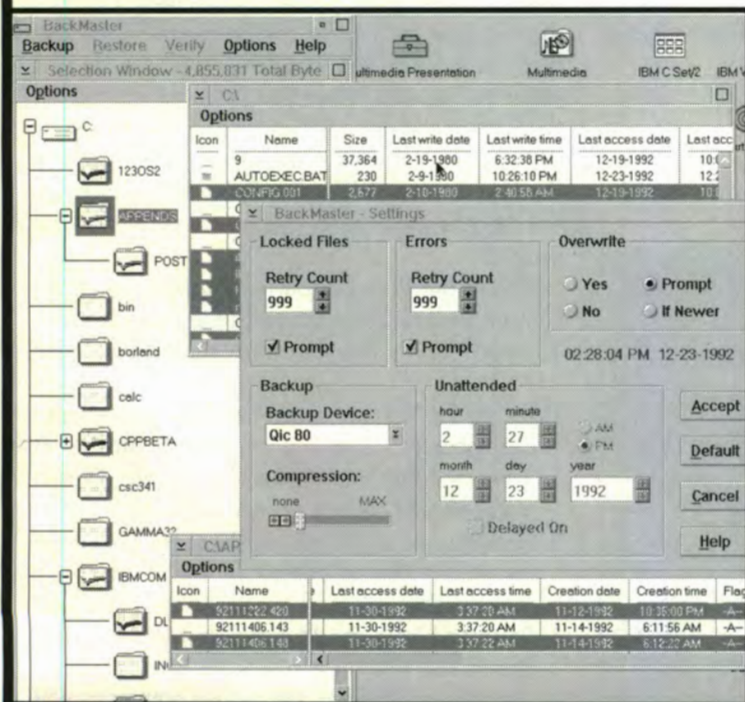
Cheatham, Paul W., David E. Reich, and Robert F.G. Robinson. "Presentation Manager Architecture," *IBM Personal Systems Developer*, Feb. 1989, pp. 33-38.

*IBM OS/2 Standard Edition Kernel and Presentation Manager* (IBM Doc. GG24-3370-00)

*Microsoft OS/2 Programmer's Reference, Volume 1*. Redmond, Wash.: Microsoft Press, 1989.



# BackMaster for OS/2 2.x



## 32-Bit Backup for OS/2

32-Bit Multi-Threaded PM Backup Utility.  
Supports FAT and HPFS File Systems.  
Handles Long Names, and Extended Attributes.  
Backup WPS, Desktop, and System Files.  
Total/Partial/Incremental/Unattended Backups.  
Backup to Floppy Disk and QIC-40/80 Tape Drives.  
Can Read QIC-40/80 DOS Tapes.  
Easily Migrate your DOS files to OS/2 using Tape.  
We Support CMS Jumbo & Generic QIC-40/80  
Runs in the Background.  
Uses Industry Standard STAC LZS Data Compression.

**Only \$79.95**

**Texas Residents add 8 1/4% Sales Tax**  
**VISA/MC/COD Welcome**

Call our BBS for a Free Demo Version

**MSR Development**

Rt 7 #6409, Nacogdoches, TX 75961  
Ph (409) 564-1862, BBS (409)560-5970





MMPM/2 provides not only multimedia device control but also easy-to-program animated buttons, sliders, and secondary windows. **BY DONALD G. JOHNSTON**

# Programming the Graphical User Interface Extensions of MMPM/2



Donald G. Johnston

**P**revious articles on MMPM/2 have covered the Media Control Interface, which allows high-level programs to work with multimedia devices with a minimal amount of programming. This article covers another major addition provided by MMPM/2, the addition of secondary windows, graphical push buttons (two-state and animated), and circular sliders. These build on the home entertainment system paradigm and help reduce the user's learning curve when faced with new multimedia applications. The stop, pause, play, and volume controls that people are used to on stereos will now appear on the computer screen.

### OPERATING THE WINDOW

When a user clicks on the Play push button, shown in Figure 1, it appears to depress into the

screen, beep, and then start animating; the three-dimensional effect and the animation are handled by MMPM/2, while the beep and the notification to start animation are part of the program covered in this article. Clicking on the Stop push button causes the same three-dimensional effect, triggers a beep of a different tone, and stops the animation of the Play push button.

An important point of the MMPM/2 support for these graphical controls is the extension to the normal keyboard control that is possible with PM objects. Pushing the Tab key jumps focus between the two groups defined (the Stop and Play push buttons make up the first group, and the circular slider the second). When you tab to the first group, the Stop push button is highlighted; use the left or right cursor control key to change the highlight to the Play button. When the Enter key is pressed, the highlighted push button appears to be pressed.

While the Play button is animating, you can control the animation rate with the circular slider. There are several ways to control the circular slider:

1. Tab to the Volume control and use the left and right cursor control keys, or click on the -/+ icons at either end of the slider range. The set value should change by increments of 1.
2. Click on one of the tick marks at the outside of the slider. The set value (and the pointer) will jump to that value.
3. Press and hold the Mouse Select button while the mouse pointer is inside the circular slider. The slider will track the mouse pointer as you



Figure 1. A secondary window





move it around the screen. The set value will increase by increments of 10 at most, or enough to catch the mouse.

This program also has a default size item in the system menu. If the window has been resized, it can be returned to its original dimensions by clicking on this item, part of the secondary window support provided by MMPM/2.

### THE TEMPLATE FOR THE SECONDARY WINDOW

The dialogue box definition file (MMPMGUI.DLG), shown in Figure 2, defines the layout (or template) of the dialogue box used as the secondary window. It is defined as a dialogue box template by the keyword DLGTEMPLATE.

The #define statements ensure that the correct

sections of the secondary window header file (SW.H from the MMPM Toolkit/2) are used to compile this resource.

The DIALOG statement defines the window. The text data MMPM/2 GUI Support appears in the title bar. The various styles (FS\_) and flags (FCF\_) define the layout of the window, delineating a sizing border, a system menu, and so on.

The CONTROL statement (defining either a WC\_GRAPHICBUTTON or a WC\_CIRCULARSLIDER) is used to add graphic controls to the dialogue box. The graphic buttons can be either two-state (Stop) or animated (Play).

The CTLDATA statement defines the control data associated with the push buttons and defines the bitmap or bitmaps, such as ID\_BMP\_STOP and ID\_BMP\_PLAYn, that are to be used. The constants used in this definition are defined in the MMPMGUI.H file,

## THE PARALLEL SOLUTION

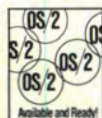


**TAPE BACKUP**  
plugs into any standard  
parallel port, supports DOS,  
OS/2, Windows, NetWare, SCO UNIX and XENIX.



• Complete with powerful solution-minded software series, including BUSS (BackUp Supervisor Software), SM (Script Manager) and SDB (SCSI Disk Backup).

• Timesaving, simultaneous backup and verify at true 11MB/min.



• 250MB-4GB capacities in native mode.

• Optical 128MB-1.2GB, removable media.

• QIC industry standard format.

• Extremely portable, weighs as little as a notebook.



**PSS PARALLEL STORAGE SOLUTIONS**

The Pacesetter in Parallel to SCSI Technology.

**INFO AND ORDERS: 1-800-998-7839**

CORPORATE HEADQUARTERS • 116 S. CENTRAL AVE • ELMSFORD, NY USA 10523 • FAX: (914) 347-4646

## Waldenbooks

### Waldenbooks® and Que Present the Official Object-Oriented Design Guide



#### Object-Oriented Interface Design Guidelines

IBM® Corporation

Uncover the secrets of interface design! This fantastic tutorial and reference describes the fundamental concepts of the Common User Access™ (CUA™) interface.

**que**

ISBN# 1-56529-170-0  
IBM Pub# SC34-4399-0  
\$29.95 USA  
Waldenbooks Item# 0967

Available at Waldenbooks®,  
Waldensoftware®, & Brentano's®

- Learn about the user interfaces and object-oriented environments
- Explore a CUA designer model
- Examine the components of the CUA interface
- Apply instructions on how to design a product with a CUA interface

To Order Direct, Call 1-800-443-7359 Dept# 768



shown in Figure 3, which is included with the `DLGINCLUDE` command.

### CONNECTING THE BITMAPS TO THE CONTROLS

The resource compiler file `MMPMGUI.RC`, shown in Figure 4, is used by the resource compiler to draw together all resources used by the `MMPMGUI.DLG` program. `RCINCLUDE` includes, as one of the resources, the dialogue box definition file in Figure 2.

The `BITMAP` statements define the file used for each symbolic reference (for example, `STOP.BMP` is used for `ID_BMP_STOP`). These bitmaps can be copied from the `DUET1` sample code subdirectory that comes with the `MMPM Toolkit/2`.

The `STRINGTABLE` statement is used to define all character strings used. Each string ("Default Size) has an ID number (`IDS_DEFAULTSIZE`) associated with it. The tilde (~) marks the character to be underlined. The ID number `IDS_DEFAULTSIZE` is defined in the `MMPMGUI.H` file as 1.

### DEFINING THE PROGRAM STRUCTURE

The C compiler uses the definition file `MMPMGUI.DEF`, shown in Figure 5, to get information about the type of program it is about to compile.

The `NAME` parameter defines this as a PM program (`WINDOWAPI`) with the name `MMPMGUI`. The `DESCRIPTION` text `MMPM/2 GUI Support` ends up in the header of the executable file. The program has one entry point (`MainProc`), used by the OS/2 operating system to direct messages meant for it.

For all `MMPM/2` programs the `STACKSIZE` should be at least 16,000 bytes.

### PROGRAMMING FOR A SECONDARY WINDOW

It is now the responsibility of your C program to first ask `MMPM/2` to create the secondary window and then control various parameters of the graphic controls and receive messages from the window. Support code that does this is shown in Figure 6.

As `#include <os2.h>` and `#define INCL_WIN` are used in standard PM programs, we use `#include <os2me.h>` and

```

/*****
 *
 * File: MMPMGUI.DLG
 *
 *****/

#define INCL_CIRCULARSLIDER
#define INCL_GRAPHICBUTTON

#include <sw.h>

DLGINCLUDE 1 "mmpmgui.h"

DLGTEMPLATE ID_DLG_MAIN LOADONCALL MOVEABLE DISCARDABLE
BEGIN
    DIALOG "MMPM/2 GUI Support", ID_DLG_MAIN, 12, 6, 148, 84,
        NOT FS_DLGBOARD | FS_SIZEBOARD | WS_VISIBLE,
        FCF_SYSMENU | FCF_TITLEBAR | FCF_MINBUTTON |
        FCF_MAXBUTTON

    BEGIN
        CONTROL        "", ID_GPB_STOP, 28, 60, 23, 14,
            WC_GRAPHICBUTTON,
            WS_GROUP | WS_TABSTOP | WS_VISIBLE
            CTLDATA GB_RESOURCE, "", 1, ID_BMP_STOP

        CONTROL        "", ID_GPB_PLAY, 85, 60, 40, 14,
            WC_GRAPHICBUTTON,
            GBS_ANIMATION | WS_VISIBLE
            CTLDATA GB_RESOURCE, "", 5, ID_BMP_PLAYO,
            ID_BMP_PLAY1, ID_BMP_PLAY2,
            ID_BMP_PLAY3, ID_BMP_PLAY4

        CONTROL        "Volume", ID_SL_VOLUME, 33, 0, 85, 58,
            WC_CIRCULARSLIDER,
            WS_GROUP | WS_TABSTOP | WS_VISIBLE

    END
END

```

Figure 2. The dialogue box template

`#define INCL_SW` to include all secondary window support provided by `MMPM/2`.

The secondary window defined in the dialogue template file is created when the `WinLoadSecondaryWindow()` function is used (instead of the `WinCreateStdWindow()` used in PM programs). This window still needs the normal anchor block (`hab`) and message queue (`hmq`)

used to receive messages. To create the `Default Size` item in the system menu, `MMPM/2` provides the `WinInsertDefaultSize()` function.

Before control is returned from the `WinLoadSecondaryWindow()` function, a `WM_INITDLG` message is sent to the dialogue procedure of the program `MainProc()`. When this message is received, the animation rate of the Play push but-



```

/*****
 *
 * File: MMPMGUI.H *
 *
 *****/

#define ID_DLG_MAIN          100
#define ID_GPB_STOP          101
#define ID_GPB_PLAY          102
#define ID_SL_VOLUME         103

#define ID_BMP_STOP          200
#define ID_BMP_PLAY0         201
#define ID_BMP_PLAY1         202
#define ID_BMP_PLAY2         203
#define ID_BMP_PLAY3         204
#define ID_BMP_PLAY4         205

#define IDS_DEFAULTSIZE      1

```

Figure 3. The program's header file

```

/*****
 *
 * File: MMPMGUI.RC *
 *
 *****/

#include <os2.h>
#include "mmpmgui.h"

RCINCLUDE mmpmgui.dlg

BITMAP ID_BMP_STOP LOADONCALL MOVEABLE DISCARDABLE STOP.BMP

BITMAP ID_BMP_PLAY0 LOADONCALL MOVEABLE DISCARDABLE PLAY0.BMP
BITMAP ID_BMP_PLAY1 LOADONCALL MOVEABLE DISCARDABLE PLAY1.BMP
BITMAP ID_BMP_PLAY2 LOADONCALL MOVEABLE DISCARDABLE PLAY2.BMP
BITMAP ID_BMP_PLAY3 LOADONCALL MOVEABLE DISCARDABLE PLAY3.BMP
BITMAP ID_BMP_PLAY4 LOADONCALL MOVEABLE DISCARDABLE PLAY4.BMP

STRINGTABLE
{
  IDS_DEFAULTSIZE
    "Default Size"
}

```

Figure 4. The resource compiler file

```

;
; File: MMPMGUI.DEF
;

NAME            MMPMGUI    WINDOWAPI

DESCRIPTION      'MMPM/2 GUI Support'
HEAPSIZE         2000
STACKSIZE        20000
EXPORTS          MainProc

```

Figure 5. The definition file

ton is set with a `GBM_SETANIMATIONRATE` message. The parameter 100 indicates that the bitmap is to be changed every 100 milliseconds ( $1/10$  second). For the circular slider, the range (0-100), increment value (10), and initial value (80) can be set with the messages `CSM_SETRANGE`, `CSM_SETINCREMENT`, and `CSM_SETVALUE`.

Watch for whenever your program sends a `CSM_SETVALUE` message to the circular slider. The slider will, in turn, send a `CSN_CHANGED` message back to the window's dialogue procedure; your program should be ready to handle this.

When a user clicks on the Play push button, MMPM/2 creates the three-dimensional effect of the button pushing into the screen and then sends a `WM_COMMAND` message to `MainProc()`. (The first parameter, `ID_GPB_PLAY`, defines which button caused the message.) The program sends a `GBM_ANIMATE` message, with the `TRUE` parameter, back to the button to start it animating. MMPM/2 will then use the five bitmaps defined in `MMPMGUI.DLG`, cycling through them every 100 milliseconds to give the appearance of movement.

Clicking on the Stop push button causes the same three-dimensional effect as that for the Play button and a `WM_COMMAND` message with the `ID_GPB_STOP` button identified. A `GBM_ANIMATE` message with the `FALSE` parameter is then sent to the Play push button to stop its animation.

The `DosBeep()` function provides user feedback and adds a little excitement to the program. It also delays the return to MMPM/2, which shows that the





graphic buttons remain down while the program processes a message. The flow, then, is as follows: MMPM/2 pushes the button in, sends the program a synchronous message, then pops the button out.

Graphics buttons under MMPM/2 also send WM\_CONTROL messages to the dialogue procedure. This is the best way to handle messages from the circular slider. All the effects of this slider are created by MMPM/2, and the program needs only handle the CSN\_CHANGED and CSN\_TRACKING parameters of the WM\_CONTROL message received from the ID\_SL\_VOLUME control. CSN\_TRACKING is sent for each increment as MMPM/2 tracks the mouse. CSN\_CHANGED is sent to provide the program with the final value set from any user action on the slider. Normally these would be used to control the volume of a multimedia device, but here they just control the animation rate of the Play push button.

All messages not explicitly handled by this program are passed on to WinDefSecondaryWindowProc(), MMPM/2's default secondary window procedure.

## REFERENCES

"Multimedia Presentation Manager/2 Programming Using Digital Audio Examples," *ITSC Technical Bulletin* (IBM Doc. GG24-3963).

"Multimedia Presentation Manager/2 Advanced Programming Techniques," *ITSC Technical Bulletin* (IBM Doc. GG24-4042).

*CUA Guide to Multimedia User Interface Design* (IBM Doc. S41G-2922)

*MMPM/2 Application Programming Guide 1.1* (IBM Doc. S71G-2221)

*MMPM/2 Programming Reference 1.1* (IBM Doc. S71G-2222)

```

/*****
 *
 * File:  MMPGUI.C
 *
 *****/

#define INCL_WIN
#define INCL_SW

#include <os2.h>
#include <os2me.h>
#include "mmpgui.h"

/* Procedure Prototype */
MRESULT EXPENTRY MainProc(HWND hwnd,
    ULONG msg, MPARAM mp1, MPARAM mp2);

/* Global Variables */
HAB      hab;
HMQ      hmq;
QMSG     qmsg;
HWND     hwndFrame, hwndPlay, hwndCirc;

INT main( VOID )
{
    CHAR  szDefaultSize[30];

    hab = WinInitialize( 0);
    hmq = WinCreateMsgQueue( hab, 0);
    hwndFrame = WinLoadSecondaryWindow(HWND_DESKTOP, HWND_DESKTOP,
        MainProc, (HMODULE) NULL, ID_DLG_MAIN, NULL);

    WinLoadString( hab, (HMODULE) NULL, IDS_DEFAULTSIZE,
        sizeof(szDefaultSize), szDefaultSize);
    WinInsertDefaultSize(hwndFrame, szDefaultSize);

    while ( WinGetMsg( hab, &qmsg, (HWND) NULL, 0, 0 ) )
        WinDispatchMsg( hab, &qmsg );

    WinDestroySecondaryWindow( hwndFrame );
    WinDestroyMsgQueue( hmq );
    WinTerminate( hab );
    return( 0);
}

/*****
 * Main Dialog Procedure *
 *****/

MRESULT EXPENTRY MainProc(HWND hwnd, ULONG msg, MPARAM mp1, MPARAM mp2)
{
    switch( msg)
    {
        case WM_INITDLG:
            hwndPlay = WinWindowFromID( hwnd, ID_GPB_PLAY);

            WinSendMsg( hwndPlay, GBM_SETANIMATIONRATE,
                MPFROMLONG(100L), NULL);
    }
}
```

Figure 6. The C support code (continued on page 85)



```

    hwndCirc = WinWindowFromID( hwnd, ID_SL_VOLUME);

    WinSendMsg( hwndCirc, CSM_SETRANGE,
        MPFROMLONG(0L), MPFROMLONG(100L));
    WinSendMsg( hwndCirc, CSM_SETINCREMENT,
        MPFROMLONG(10L), MPFROMLONG(2L));
    WinSendMsg( hwndCirc, CSM_SETVALUE,
        MPFROMLONG(80L), NULL);

    break;

case WM_COMMAND:
    switch (SHORT1FROMMP( mp1))
    {
        case ID_GPB_PLAY:
            DosBeep( 200, 400);

            WinSendMsg( hwndPlay, GBM_ANIMATE,
                MPFROMSHORT(TRUE), NULL);

            break;

        case ID_GPB_STOP:
            DosBeep( 550, 800);

            WinSendMsg( hwndPlay, GBM_ANIMATE,
                MPFROMSHORT(FALSE), NULL);

            break;
    }
    break;

case WM_CONTROL:
    switch (SHORT1FROMMP( mp1))
    {
        case ID_SL_VOLUME:
            switch (SHORT2FROMMP( mp1))
            {
                case CSN_CHANGED:
                    WinSendMsg( hwndPlay, GBM_SETANIMATIONRATE,
                        MPFROMLONG(150L - SHORT1FROMMP( mp2)), NULL);

                    break;

                case CSN_TRACKING:
                    DosBeep( 600 + 2 * SHORT1FROMMP( mp2), 200);

                    break;
            }
        }
    }
    break;

default:
    return( WinDefSecondaryWindowProc( hwnd, msg, mp1, mp2));
}
}

```

**Don Johnston**, IBM Canada Ltd., 600-1801 Hamilton St., Regina, Sask., Canada, S4P-4B4. Johnston is an advisory systems engineer currently on assignment from Canada with the International Technical Support Center (ITSC) in Boca Raton, Florida, where he provides third-level technical support for multimedia and premium desktop systems to countries outside the United States. Johnston is a registered professional engineer in electrical engineering.



Figure 6. The C support code (continued from page 84)



OS/2 device drivers must handle address conversion in cases where the operating system cannot do so. Using an application and device driver `DevIOctl` example, this article demonstrates how to determine when address conversion is necessary and how to convert real to virtual addresses. **BY FRANK J. SCHROEDER**

# Converting Real Addresses to Virtual Addresses in OS/2 2.x



Frank J. Schroeder

**O**ne obstacle OS/2 1.x device drivers may need to overcome when running under the virtual memory environment of OS/2 2.x is how to correctly convert a DOS application's real memory pointer, passed in the command or data packets on a `DevIOctl` API, to its virtual equivalent. This article explains why the operating system does not convert embedded pointers for the device driver. It provides a device-driver example demonstrating how to do the address conversion, as well as DOS and OS/2 16-bit application code showing how applications pass embedded pointers on `DevIOctl` calls. The OS/2 device driver's `DevIOctl` routine shows the necessary calculations to convert an embedded pointer prior to accessing an appli-

(`DevIOctl`) that enables applications to communicate with device drivers. Every device driver provides unique functions to control its device; usually these functions set a device-specific feature or return the current device-specific feature setting. Each device driver defines the functions it supports, the parameters needed to perform the functions, and, within the confines of the `DevIOctl` API, how the parameters are passed to the device driver.

A DOS application, when started in a DOS session of the OS/2 2.x operating system, executes in the Virtual 86 (V86) mode of the Intel processor. The Intel processor in V86 mode uses a segment and offset value to directly address physical memory. The segment and

*Since an OS/2 device driver executes in protected mode, any real address must be converted to its appropriate virtual address prior to accessing the data at that location.*

cation's buffer. This article assumes the reader has general knowledge of 8086 assembly language and OS/2 device drivers.

## INTRODUCTION

The DOS and OS/2 operating systems provide an API called device input/output control

offset address is also referred to as a real address. An OS/2 application, when started in an OS/2 session of the OS/2 2.x operating system, executes in the protected mode of the Intel processor. The Intel processor in protected mode uses a selector and offset value to translate the address to physical memory. The selec-



tor and offset address is also referred to as a virtual address.

A DOS application, when executing in V86 mode, passes a real address on API calls to the operating system. Since an OS/2 device driver executes in protected mode, any real address must be converted to its appropriate virtual address prior to accessing the data at that location.

### DevIOctl API

The DevIOctl API defines a set of parameters that can be customized to each device driver. The parameters passed by a DOS application through the DOS version of the DevIOctl API (int 21h, AH=44h) are identical to the set of parameters passed by 16-bit OS/2 applications through the OS/2 version of the DevIOctl API (DosDevIOctl). The category-code parameter is a numeric value that categorizes the type of device driver being called (such as parallel port, serial port, mouse, or disk). The function-code parameter is a numeric value that defines the function the device driver is requested to perform. A set of reserved, operating-system-specific category and function codes are defined in Appendix C of the *DOS 5.0 Technical Reference* and chapter 18 of the *OS/2 2.0 Physical Device Driver Reference*. (See the References section for details.)

In addition to the category- and function-code parameters, the DevIOctl API defines two parameters (the command and data packets) that pass function-specific parameters to the device driver. The addresses of the command and data packets are passed to the device driver on the DevIOctl API. If the command and data packet pointers are V86 mode addresses, the operating system converts these addresses to their appropriate virtual addresses prior to calling the device driver.

The command and data packets contain parameters in a format predefined by the device driver. If a device driver defines an embedded pointer as a parameter passed in the command or data packet, this far pointer cannot be converted by the operating system into its virtual address, because the application and the device driver are the only ones that under-

stand the format of these packets. It is the device driver's responsibility to determine whether the process calling the device driver is a V86 mode process and convert the embedded pointer to its appropriate virtual address, so the device driver can complete the requested function.

DOS applications that use the DOS protected-mode interface (DPMI) must convert their embedded pointers to the segment and offset format prior to issuing a DevIOctl API. The OS/2 device driver treats DPMI applications as DOS applications, assumes the embedded pointer is a real address, and tries to convert it accordingly. If the DPMI application passes a protect-

|            |                      |   |                                 |
|------------|----------------------|---|---------------------------------|
| v86Mode    | STRUC                |   |                                 |
| v86_off    | dw                   | ? | ; v86Mode Address Offset        |
| v86_seg    | dw                   | ? | ; v86Mode Address Segment       |
| v86_len    | dw                   | ? | ; v86Mode Buffer Length         |
| v86Mode    | ENDS                 |   |                                 |
| datapkt db | SIZE v86Mode DUP (0) |   | ; devioctl buffer - data packet |
| cmdpkt db  | 0                    |   | ; devioctl buffer - cmd packet  |
| buffer db  | "PARMS_GO_HERE"      |   | ; buffer                        |
| BUFLen EQU | \$ - buffer          |   | ; size of buffer                |

Figure 1. DevIOctl data structures

|     |                          |  |                        |
|-----|--------------------------|--|------------------------|
|     | ; setup ioctl parameters |  |                        |
| lea | bx,datapkt               |  | ; ds:bx -> data packet |
| mov | ax,OFFSET buffer         |  | ; get buffer offset    |
| mov | [bx].v86_off,ax          |  | ; save buffer offset   |
| mov | [bx].v86_seg,ds          |  | ; save buffer segment  |
| mov | [bx].v86_len,BUFLen      |  | ; save buffer length   |

Figure 2. DevIOctl data packet initialization

ed-mode address, the device driver performs the address conversion when it should not.

The 16-bit OS/2 DevIOctl2 API and the 32-bit OS/2 DevIOctl API also pass, as parameters, the lengths of the command and data packets and the addresses of these variables, so that the device driver can update the length of the com-



mand and data packets returned to the application. A device driver must be at least level-two to receive the command and data packet lengths. The device-driver level is specified within the device-driver header found at offset 0 of the device driver's first data segment. The example described in this article uses 16-bit instructions and the 16-bit version of the `DevIOctl` API, so it does not use the command and data packet length parameters.

### OPEN AND CLOSE REQUESTS

Before issuing a `DevIOctl` request, the DOS or OS/2 application must first issue an `Open` request to the device driver. The `Open` API assigns and returns a handle parameter to the application that must be passed on subsequent requests to the device driver. The handle is used by the operating system as a method of retrieving and storing information to its data structure regarding communication with the device driver. The final step to terminate communication with device drivers is issuing a `Close` API to free the entry in the operating-system data structure. Because the open and close procedures vary according to the device being opened, they are not shown here.

### DevIOctl DATA STRUCTURES

The DOS and OS/2 application must allocate command and data-packet storage to be used when passing information to the device driver, as shown in Figure 1. The command packet is not used for parameter passing in this example and consequently contains only a null (00h) byte. The amount of data-packet storage and the format of the data packet are defined by the `V86Mode` structure. The data packet allocates storage for a segment or selector (`V86_seg`) and an offset (`V86_off`) to address a separate buffer containing the parameters being passed to the device driver. The data packet also

; issue ioctl request

```
mov    ah,44h                ; ioctl function call
mov    al,0Ch                ; handle based call
mov    dx,bx                 ; ds:dx -> data packet
mov    bx,handle             ; handle
mov    ch,80h                ; category code - user type
mov    cl,40h                ; function code - send data
mov    si,ds                  ; si:di -> cmd packet
lea    di,cmdpkt             ; call device driver
int     21h                  ; jmp if error
jc     error
```

Figure 3. `DevIOctl` from a DOS application

; issue ioctl request

```
push    ds
lea     dx,datapkt           ; data packet
push    dx
push    ds
lea     dx,cmdpkt            ; command packet
push    dx
push    40h                  ; function code - send data
push    80h                  ; category code - user type
push    handle               ; handle
call    DosDevIOctl          ; call device driver
or      ax,ax                ; jmp if error
jnz     error
```

Figure 4. `DevIOctl` from an OS/2 application

.286C

DSEG SEGMENT PUBLIC 'data'

; Generic IOctl Request Packet Structure Definition

```
ReqPacket  STRUC
RPLength   db    ?      ; Request Packet Length
RPUnitCode db    ?      ; Request Packet Unit Code
RPCommand  db    ?      ; Request Packet Command Code
RPStatus   dw    ?      ; Request Packet Return Status
RPReserved dd    ?      ; Request Packet Reserved
RPQueue    dd    ?      ; Request Packet Queue Linkage
Category   db    ?      ; Category Code
Function    db    ?      ; Function Code
GIOParmPack dd    ?      ; PTR to Parm Packet
GIODataPack dd    ?      ; PTR To Data Packet
GIOSFN     dw    ?      ; System File Number
```

Figure 5. Device driver real-to-virtual address conversion routine



contains storage for the length of the separate buffer being passed to the device driver (V86\_len). Called buffer, it contains the string PARS\_ GO\_HERE.

### **DevIOctl DATA PACKET INITIALIZATION**

Next, the application initializes the parameters being passed within the data packet of the DevIOctl API to the device driver, as shown in Figure 2. This code is identical, regardless of whether the application is a DOS (real or V86 mode) or an OS/2 (protected-mode) application. The offset portion of the address of the buffer is stored in the data packet, followed by the segment portion (real-mode application) or selector portion (protected-mode application) of the buffer address, and, finally, the length of the buffer. The DS register contains the mode-specific application data segment or selector value initialized by the operating system before the application begins execution.

The category code selected (80h) corresponds to the first category code not reserved for operating-system use. The function code chosen (40h) corresponds to the first function code available to send data to the device driver. The category and function codes selected for use in this example are somewhat arbitrary; however, they do comply with the rules for using these parameters defined within the *OS/2 2.0 Physical Device Driver Reference*. No reference to restricted use of these parameters could be found within the *DOS 5.0 Technical Reference* manual.

### **CALLING DevIOctl FROM A DOS APPLICATION**

Parameter-passing to the DOS version of the DevIOctl API is accomplished by loading the general-purpose registers with the required parameters and issuing an interrupt 21h instruction to call the device driver to perform the function, as in Figure 3. The handle, category code, function code, and command and

data packet addresses are loaded into the appropriate registers prior to issuing the request. The command and data packet addresses are in the V86 mode format and are converted automatically by the operating sys-

tem into their virtual addresses before being passed to the device driver. If an error occurs during the attempt to process the DevIOctl request, the request returns with the carry flag set and the error code in

## **Brief's performance is over. Meet the new star.**

### **Introducing the RimStar Programmer's Editor V2.0**

If you're looking for a fully configurable, professional OS/2 PM programmer's editor to replace your current text-mode editor, we have what you've been looking for. We know changing editors can be a major hassle. You don't want to relearn a new set of keystrokes no matter how good the product. Well, you don't have to - we have Brief, Epsilon, Multi-Edit, SlickEdit, PWB and Borland IDE keyboard mappings waiting for you. If you're not using one of these, let us know and we will create the mapping you need.

Flexibility, customization and ease-of-use are key requirements for any professional programmer's editor. The RimStar Programmer's Editor does more than just meet these requirements, it raises them to a whole new level.

It has all the features you have come to expect, plus special features designed to anticipate your needs and increase your productivity. Features like a 'C' source browser that will eliminate those time consuming searches for functions, global variables, typedefs, and macros by providing you instant positioning to both definitions and references. Using our powerful C macro language you can customize your programming environment to suit your individual needs.

#### **The RimStar Programmer's Editor features**

- ✓ Complete C macro language
- ✓ Auto-indent & template editing
- ✓ No limits on files, file size, or windows
- ✓ Line lengths to 16K
- ✓ Timed auto save
- ✓ Access PDK help for function under cursor
- ✓ Configurable tool bar
- ✓ Customizable menus
- ✓ Complete on-line help
- ✓ Integrates with Workframe/2
- ✓ Save and restore state between sessions
- ✓ Source browser for C
- ✓ Unlimited undo and redo
- ✓ Multi-buffer regular expression search and replace
- ✓ Compile and jump to errors
- ✓ Brace matching
- ✓ Column, line and block selection, search and replace
- ✓ Block indent/outdent
- ✓ Keystroke record/playback
- ✓ Multi-threaded for no waiting
- ✓ Import/export to system clipboard
- ✓ OS/2 2.x 32 bit PM Multi-document interface

**Brief users upgrade now for only \$99.00\***

**To order call (603) 778-2500.**

**90 day money-back guarantee.**

#### **RimStar Technology, Inc.**

91 Halls Mill Road  
Newfields, NH 03856-0938

"My copy of Brief has been permanently retired. Keep up the good work!" -AL

\*Offer applies to other editors and expires 12/31/93. Regular list price \$249. All products and company names are trademarks or registered trademarks of their respective holders.



the **AX** register. The DOS **DevIOctl** parameter-passing convention can be found in Appendix C of the *DOS 5.0 Technical Reference*.

### CALLING **DevIOctl** FROM AN OS/2 APPLICATION

Parameter-passing to the OS/2 version of the **DevIOctl** API is accomplished by pushing the required parameters onto the stack and calling the device driver to perform the function, as shown in Figure 4. The handle, category code, function code, and command and data packet addresses are pushed onto the stack prior to issuing the dynamic-link call. The command and data packet addresses are already in their virtual format and do not need to be further

*Access to an application's data at interrupt time requires a GDT selector.*

processed before being passed to the device driver. If an error occurs during the attempt to process the **DevIOctl** request, the request returns with an error code in the **AX** register. If the **AX** register contains 0, then no error occurred. The OS/2 **DevIOctl** parameter-passing convention can be found in the *OS/2 2.0 Control Program Programming Reference*.

### DEVICE DRIVER **DevIOctl** REQUEST PACKET

The operating system copies the parameters that are either in the general-purpose registers or on the stack into a request-packet control block that is passed to the device driver. The request-packet address is passed to the device driver in the **ES** and **BX** registers, and the device driver is called at its strategy entry point to process the request. The device driver validates the parameters and

```

GIOParamLen    dw    ?    ; Parm Packet Length
GIODataLen     dw    ?    ; Data Packet Length
ReqPacket      ENDS

; RPStatus Return Codes

REQDONE        EQU     0100h ; REQUEST IS DONE, NO ERROR, NO RET CODE
ERROR_INVALID_PARAMETER EQU 8113h ; REQUEST IS DONE, ERROR, RETURN CODE

; Convert's Generic IOctl Data Packet Structure Definition

v86Mode        STRUC
v86_off        dw    ?    ; v86Mode Address Offset
v86_seg        dw    ?    ; v86Mode Address Segment
v86_len        dw    ?    ; v86Mode Buffer Length
v86Mode        ENDS

; Device Helper Function Codes

DevHlp_VerifyAccess EQU 39 ; 27 Verify access to memory
DevHlp_AllocGDTSelector EQU 45 ; 2D Allocate GDT selectors
DevHlp_GetDOSVar EQU 36 ; 24 Return pointer to DOS variable
DevHlp_FreeGDTSelector EQU 83 ; 53 Free allocated GDT selectors
DevHlp_LinToGDTSelector EQU 92 ; 5C Convert linear address to virtual

; Local Info Segment Structure Definition

LocalInfoSegmnt STRUC
LIS_CurProcID   DW    ?    ; Current Process ID
LIS_ParProcID   DW    ?    ; Parent Process ID
LIS_CurThrdPri  DW    ?    ; Current Thread Priority
LIS_CurThrdID   DW    ?    ; Current Thread ID
LIS_CurSessID   DW    ?    ; Current Session ID
LIS_ProcStatus  DB    ?    ; Process Status
LIS_reserved1   DB    ?    ; Reserved
LIS_FgndProc    DW    ?    ; Current Process in Foreground
LIS_ProcType    DB    ?    ; Type of Process
LIS_reserved2   DB    ?    ; Reserved
LIS_EnvironSel  DW    ?    ; Environment Selector
LIS_CommandLine DW    ?    ; Command Line Offset
LIS_DSegLen     DW    ?    ; Length in Bytes of Data Segment
LIS_StackSize   DW    ?    ; STACKSIZE in Bytes of .EXE file
LIS_HeapSize    DW    ?    ; HEAPSIZE in Bytes of .EXE file
LIS_ModHandle   DW    ?    ; Module Handle of the Application
LIS_DSegHan     DW    ?    ; Data Segment Handle of Application
LocalInfoSegmnt ENDS

; Local Info Segment ProcType Definitions

LIS_PT_FULLSCRN EQU 0
LIS_PT_REALMODE EQU 1
LIS_PT_VIOWIN   EQU 2
LIS_PT_PRESMGR  EQU 3

```

Figure 5. Device driver real-to-virtual address conversion routine (continued on page 91)



```

LIS_PT_DETACHED EQU    4

; Local Info Segment Address

LIS_Addr          STRUC
LIS_Addr_Off      DW    ?      ; Local Info Segment Offset
LIS_Addr_Sel      DW    ?      ; Local Info Segment Selector
LIS_Addr          ENDS

; Misc Parameters needed for Device Helper Services

LOCINFOSEG        EQU    2      ; System Local Info Segment - GetDOSVar
RACCESS           EQU    0      ; Read Access - VerifyAccess
GDTALLOCATED      EQU    1      ; Flag Equates

; Global Data Area

PUBLIC device_help
device_help        dd    0      ; Device Helper Services Pointer
localinfoseg       dd    0      ; Local Info Segment Pointer
reqpkt            dd    0      ; Request Packet Pointer
gdtssel           dw    0      ; GDT Selector
flags             dw    0      ; Error Processing Flags

DSEG      ENDS

CSEG      SEGMENT PUBLIC 'code'
          ASSUME CS:CSEG,DS:DSEG,ES:NOTHING,SS:NOTHING

; ROUTINE NAME:      Convert.  Converts V86 address to GDT selector.
;
; FUNCTION:          The convert routine converts a V86 address (segment:
;                    offset) passed in a buffer within a Generic IOCtl
;                    API call into a GDT selector that the device driver
;                    can use to access this memory at task or interrupt
;                    time.
;
; NOTE:              OS/2 kernel cannot convert an imbedded pointer
;                    because it does not know about them.
;
; ENTRY:             es:bx -> to Generic IOCtl request packet
;                    ds      our data segment
;
; EXIT:              all other registers are not preserved
;
          PUBLIC convert
convert proc        near
and                flags,NOT GDTALLOCATED      ; set flag off

          mov     WORD PTR reqpkt,bx           ; save req pkt offset
          mov     WORD PTR reqpkt+2,es         ; save req pkt selector

          ; GET POINTER TO LOCAL INFO SEG

```

calls the appropriate device-driver routine to service the DevIOCtl request specified by the function-code parameter.

The operating system provides services to device drivers through the device helper (DevHlp) entry point. The DevHlp entry-point address is passed to the device driver when the device driver is called to initialize. The DevHlp address must be saved by the device driver for future use. Prior to calling a DevHlp entry point, a device driver loads the DL register with the DevHlp function to be performed and the general-purpose registers with any required parameters. If an error occurs during the processing of a DevHlp request, the carry flag is set, and the AX register returns the error code.

Each DevHlp service is available to a device driver only during certain times. Initialization, task, and interrupt are the three time periods when a device driver can execute and when these DevHlp services may be available. Initialization time occurs when the device driver is called to initialize itself. Task time is the period when the device driver services requests entered through its strategy entry point. Interrupt time occurs when the device driver is called at its hardware-interrupt entry point to service its device.

Selector addresses found in the local descriptor table are not available to device drivers at interrupt time. If a device driver needs to access an application's data area at interrupt time, the data area must be addressed by a GDT selector found in the global descriptor table, the application must be in context (foreground), and the device driver needs to lock the application's data area into memory. The application does not need to be in context if the address is made HVDM relative. To be made HVDM relative, the handle to the VDM must be added to the linear address. The device driver uses the DevHlp lock service to lock

Figure 5. Device driver real-to-virtual address conversion routine (continued on page 92)



the data area and the corresponding DevHlp unlock service to unlock it once access is complete. The lock request should be issued after the device driver verifies access to the segment and prior to blocking the application's thread. The DevHlp VMlock service can be used if the application's data area must be physically contiguous or within the first 16 MB of memory. An alternative is to copy the data at task time from the application's data

area to the device driver's data segment; however, this approach has performance implications. Obviously, OS/2 device drivers have access to their data segments at interrupt time.

### DETECTING A V86 MODE PROCESS

The operating system converts the DevIOctl command and data packet addresses to their virtual addresses, but it does not know about any embedded addresses stored within the command and data packets. It is the responsibility of the device driver to convert any embedded address in a

command or data packet to its virtual address. If the device driver fails to convert the V86 address or converts it incorrectly, the device driver may attempt to access the memory by loading a segment register with a V86 address while the processor is in protected mode. (The Intel processor is in protected mode when running an OS/2 device driver and the processor does an address translation on the contents of the segment register.) If a V86 address is stored

*It is the responsibility of the device driver to convert any embedded address in a command or data packet to its virtual address.*

```

mov     al,LOCINFOSEG           ; get local infoseg addr
mov     dl,DevHlp_GetDOSVar     ; get dos var function
call    DWORD PTR [device_help] ; call device help
jnc     convert1                ; save infoseg addr
jmp     error                   ; error, exit

convert1:
mov     es,ax                   ; es:bx -> infoseg addr
mov     ax,es:[bx].LIS_Addr_Off ; get infoseg offset
mov     WORD PTR localinfoseg,ax ; save infoseg offset
mov     ax,es:[bx].LIS_Addr_Sel ; get infoseg selector
mov     WORD PTR localinfoseg+2,ax ; save infoseg selector
les     bx,localinfoseg         ; es:bx -> localinfoseg

; DETERMINE IF V86 MODE PROCESS (AND CONSEQUENTLY A V86 MODE ADDRESS)

cmp     es:[bx].LIS_ProcType,LIS_PT_REALMODE ; if not v86mode
; process then
jne     verify                  ; no conversion

; ALLOCATE A GDT SELECTOR

push    ds
pop     es                       ; es = our data segment
lea     di,gdtssel              ; es:di -> gdt variable
mov     cx,1                     ; allocate 1 selector
mov     dl,DevHlp_AllocGDTSelector ; alloc gdt sel function
call    DWORD PTR [device_help] ; call device help
jc      error                   ; error, exit
or      flags,GDTALLOCATED      ; set flag on

; CONVERT V86 ADDRESS TO LINEAR ADDRESS

.386c
les     bx,reqpkt               ; es:bx -> req pkt
les     bx,es:[bx].GIODataPack  ; es:bx -> data packet
movzx   edx,WORD PTR es:[bx].v86_seg ; v86mode segment
shl     edx,4                   ; paragraph align
movzx   eax,WORD PTR es:[bx].v86_off ; eax = offset, add
add     edx,eax                 ; edx -> data buffer

; MAP LINEAR ADDRESS TO GDT SELECTOR

mov     ax,gdtssel              ; get gdt selector
movzx   ecx,WORD PTR es:[bx].v86_len ; length of data buf
mov     ebx,edx                 ; ebx = linear addr

.286c
mov     dl,DevHlp_LinToGDTSelector ; map linear to gdt
call    DWORD PTR [device_help] ; call device help
jc      error                   ; if error, exit
xor     di,di                   ; clear offset
jmp     SHORT verify1

```

Figure 5. Device driver real-to-virtual address conversion routine (continued on page 94)



in the segment register, an incorrect-address translation occurs and an access to that memory results in the wrong physical memory or causes a protection violation.

The steps necessary to convert an embedded V86 address to a virtual address usable by the device driver are demonstrated in Figure 5. The first step is to determine if the application making the `DevIOctl` request is a V86 mode process. This step is accomplished by obtaining the pointer to the local information segment (local info seg) and checking the process-type field. If the application is a V86 mode process, the embedded pointer must be converted to a virtual address.

The local info seg address is obtained by calling the `DevHlp GetDOSVar` service. The `GetDOSVar` service usually is requested only once, in the device-driver initialization routine, and its address is saved in the device driver's data segment and retrieved when needed.

#### **CONVERTING AN EMBEDDED DevIOctl ADDRESS**

Once it has been determined that the address requires conversion, the device driver needs to:

- Allocate a GDT selector
- Convert the V86 address to its linear address
- Map the linear address to the GDT selector
- Verify access to the buffer addressed by the GDT selector
- Access the buffer
- Free the GDT selector.

To convert the V86 mode address to a virtual address, the device driver must allocate a GDT selector by calling the `DevHlp AllocGDTSelector` service. The GDT selector is returned to the device driver in its storage addressed by the `ES` and `DI` registers. A GDT selector is a virtual address allocated in the global descriptor table and is available to the device

driver at task and interrupt time. The global descriptor table is used by the processor to perform virtual-to-physical-address translations.

Next the device driver converts the V86 mode segment and offset

address into a linear address by shifting the segment value left by four bits to obtain a paragraph address and adding the offset value. A paragraph address is a real address on a 16-byte boundary.

**The GammaTech Utilities** are designed to enhance and complement your OS/2 system. These utilities provide the ability to perform vital maintenance and recovery operations easily without extensive technical knowledge. Use **GammaTech Utilities** for your critical data or you might lose it.

# LOSE IT

## GammaTech Utilities

## or for OS/2

- Undelete Erased Files
- Optimize HPFS and FAT Volumes
- Backup INI and Desktop Files
- Recover HPFS Volumes
- Protect and Backup Boot Sectors
- Identify and Mark Bad Sectors
- Reconstruct Boot Sectors
- Protect/Lock Files
- Permanently Erase Sensitive Data
- Mass Delete Selected Files
- Sort FAT Directories
- View and Edit Selected Files
- Disk Sector Edit (ASCII or Hex)
- Display Volume Information
- Alter File Attributes
- Add Comments to Files
- Display File Fragmentation
- Display Directory Information
- SAA/CUA Compliant PM Interface

# LOSE IT

**For OS/2 Version 2**

VISA, MasterCard, American Express, C.O.D. • Overnight Service Available

**(405) 947-8080 • Fax (405) 632-6537**

**SofTouch Systems, Inc., Workstation Division**

1300 S Meridian, Suite 600, Oklahoma City, Oklahoma 73108





Once the linear address has been calculated, the device driver maps the linear address to the GDT selector by calling the `DevHlp LinToGDTSelector` service. On completion of the `LinToGDTSelector` call, the previously allocated GDT selector is mapped to the physical memory pointed to by the embedded V86 mode segment and offset address. Notice that the `LinToGDTSelector` service requires parameters to be passed in 32-bit registers, even though the OS/2 device-driver model is a 16-bit model. The kernel memory-management routine that

*The **GetDOSVar** service is usually requested only once, and its address is saved in the device driver's data segment and retrieved when needed.*

performs the `LinToGDTSelector` service is written using 32-bit instructions and requires 32-bit parameters.

With the GDT selector, the device driver must verify that it may access the memory before actually accessing it. If, for example, the memory permission is set to read-only, and the device driver tries to write into it without first checking its access rights to the memory, a protection violation occurs. Since the device driver runs at ring 0 (the most protected code), a protection violation at this level results in a system halt. To prevent a protection violation, the device driver must verify access to the memory by calling the `DevHlp VerifyAccess` service before accessing it.

An OS/2 application issuing a `DevIOCTL` API executes in protected

```

; VERIFY ACCESS TO EITHER CONVERTED GDT SELECTOR AND OFFSET ADDRESS
; OR SELECTOR AND OFFSET ADDRESS PASSED VIA GENERIC IOCTL REQUEST

verify:
    les     bx,reqpkt                ; es:bx -> req pkt
    les     bx,es:[bx].GIODataPack    ; es:bx -> data packet
    mov     di,WORD PTR es:[bx].v86_off ; get v86 mode offset
    mov     ax,WORD PTR es:[bx].v86_seg ; get v86 mode segment
    mov     cx,es:[bx].v86_len        ; get length of data buf

verify1:
    mov     dh,RACCESS                ; need read access
    mov     dl,DevHlp_VerifyAccess    ; verify access func
    call    DWORD PTR [device_help]   ; call device help
    jc      error                     ; if error, exit

; CAN NOW DO WORK ON DATA AT ADDRESS IN AX:DI
dowork:

    jmp     SHORT exit                ; exit

; PERFORM ERROR HANDLING HERE
error:
    les     bx,reqpkt                ; es:bx -> req pkt
    mov     es:[bx].RPStatus,ERROR_INVALID_PARAMETER ; set return code
    test    flags,GDTALLOCATED        ; if gdt not allocated
    jz      exit1                     ; then just exit
    jmp     SHORT cleanup              ; else free selector

; PERFORM EXIT PROCESSING HERE
exit:
    les     bx,reqpkt                ; es:bx -> req pkt
    mov     es:[bx].RPStatus,REQDONE ; set return code

cleanup:
    les     bx,localinfoseg           ; get local info seg
    cmp     es:[bx].LIS_ProcType,LIS_PT_REALMODE ; if process != v86mode
    jne     exit1                     ; exit

; FREE GDT SELECTOR

    mov     ax,gdt sel               ; get gdt selector
    mov     dl,DevHlp_FreeGDTSelector ; free gdt sel func
    call    DWORD PTR [device_help]   ; call device help
    mov     gdt sel,0                ; clear gdt selector

exit1:
    ret
convert endp

CSEG     ENDS
END

```

Figure 5. Device driver real-to-virtual address conversion routine (continued from page 92)



mode; its embedded selector and offset address is already a virtual address and does not need address conversion. However, the device driver is required to verify access on the protect-mode address before accessing the memory. Once the code determines that the process is not a V86 mode process, instruction execution continues at the `DevHlp VerifyAccess` call.

The device driver uses either the GDT selector (with a zero offset) or the protect-mode selector and offset

to access the buffer containing the string `PARMS_GO_HERE`. When the device driver has completed its use of the buffer and, therefore, the GDT selector mapped to the buffer, it must free up the GDT selector by issuing a `DevHlp FreeGDTSelector` call. Of course, the `FreeGDTSelector` service only needs to be issued if the process is a V86 mode process and a GDT selector was previously allocated.

Some device-driver developers allocate GDT selectors when the device driver is called to initialize

and free GDT selectors when the device driver is called to deinstall. The benefit of this approach is the need to call the allocate and free GDT `DevHlp` services only once instead of each time a `DevIOCTL` call is made.

If an error occurred during the processing of a `DevHlp` service, the device driver returns an appropriate return code in the status field of the kernel request packet. This return code is returned to the DOS or OS/2 application in the `AX` register.

## REFERENCES

DOS 5.0 Technical Reference (IBM Doc. S91F-8614-00)

OS/2 2.0 Technical Library, Physical Device Driver Reference (IBM Doc. S10G-6266-00)

OS/2 2.0 Technical Library, Control Program Programming Reference (IBM Doc. S10G-6263-00)

Mastrianni, Steven J. *Writing OS/2 2.0 Device Drivers in C*, New York: Van Nostrand Reinhold (ISBN 0-442-01141-5)

**Frank J. Schroeder**, IBM Personal Software Products Division, 1000 N.W. 51st St., Boca Raton, Fla. 33431. Schroeder is a staff programmer in the OS/2 device drivers and multiple virtual DOS machines department. He has published articles in *OS/2 2.x Notebook*, *OS/2 Developer*, and *IBM Personal Systems Technical Solutions*. He earned a bachelor's of technology degree in computer science from Rochester Institute of Technology, New York, and an M.B.A. from the University of Miami, Fla.





Introducing  
The Developer  
Connection for OS/2

# Development's

They're all here. All the tools, tips and techniques you need to send your OS/2® development rocketing up the charts. Brought to you by the original artists: IBM's own OS/2 development team.



Join The Developer Connection for OS/2™, and you'll receive the most timely and extensive information available to the OS/2 community.

Four times a year, you'll get a CD packed with the latest productivity tools, utilities and sample programs from IBM and others. A powerful browser and easy, graphical user interface let you locate any topic instantly in our comprehensive technical library.

Each CD includes the latest releases of smashes like "The Developer's Toolkit for OS/2," "Multimedia Presentation Manager/2 Toolkit," and "Pen for OS/2 Toolkit." Plus, you also get pre-release versions of many IBM products, operating systems, internal development tools, product demos, bit maps—you name it.

But wait, there's more. You also receive



# greatest

The Developer Connection News, our newsletter filled with information about the latest OS/2 developments, new products, a Q&A column and much more. Plus, you get access to the Developer Connection forum on CompuServe™,\* where you can talk directly to the experts.

And here's music to your ears: Buy a year's subscription to The Developer Connection for OS/2 before November 30, 1993 and pay only \$149—a savings of \$50 off the regular rate. Call 1 800 6DEVCON today, and start producing some hits of your own.

# hits.

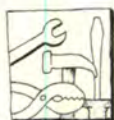
## Operate at a higher level.™



\*CompuServe membership is required. IBM and OS/2 are registered trademarks and The Developer Connection for OS/2 and "Operate at a higher level" are trademarks of International Business Machines Corporation. CompuServe is a trademark of CompuServe Incorporated. ©1993 IBM Corp.



## More REXX, GUI, SOM, And Other Tools



### TOOLS

#### **PeerLogic PIPES Platform for IBM OS/2**

**2.1.** PeerLogic's PIPES Platform is a peer-to-peer distributed computing product for developing client/server applications. Platform independent, PIPES Platform is based on a symmetrical, message-passing architecture that allows users to integrate PC LANs and SNA networks; TCP/IP and NetWare; OS/2, Windows, MVS, and UNIX at a single site or multiple remote locations.

PeerLogic Inc.

Phone: (415) 626-4545

Fax: (415) 626-4710

**Quadron qCF, qX25, qLAPB, and qCOM.** qCF, qX25, and qLAPB are OS/2 2.x toolkits for the development of communication software for IBM's ARTIC coprocessor cards. qCOM is a related turnkey product that provides multiple standard asynchronous OS/2 COM ports while offloading interrupt processing to an ARTIC card. Quadron's line of communication software development tools are used in telephone network control, financial transaction, airline reservation, process control, point-of-sale, and other communication applications.

Quadron Service Corp.

Phone: (805) 966-6424

Fax: (805) 966-7630

**Popkin SA Project Documentation Facility.** The SA PDF provides users with desktop capabilities for creating complete analysis and design documents. It facilitates the creation and management of requirement, design, and system-level documentation consistent with the System Architect CASE tool.

Popkin Software & Systems Inc.

Phone: (212) 571-3434

Fax: (212) 571-3436

**WATCOM VX-REXX.** VX-REXX is a visual solution builder for developing OS/2 2.x GUI applications. VX-REXX Operates with OS/2's REXX, which means total availability of system APIs, and includes a project management facility, visual GUI form designer, and interactive source level debugger.

WATCOM International Corp.

Phone: (800) 265-4555 or (519) 886-3700

Fax: (519) 747-4971

**ForeFront Software LAN Configuration Facility.** LCF allows for automated installation of OS/2 and applications over a LAN. Software is stored in a central library on a server where it can be accessed by users system-wide. LCF was developed and used by the Royal Bank of Canada for mission critical applications.

ForeFront Software Inc.

Phone: (403) 531-2160

Fax: (403) 270-0372

**Gpf Systems Gpf (GUI Programming Facility).** The new version of Gpf generates native OS/2 2.1 Presentation Manager WorkPlace Shell and Windows 3.1 interface code from a single design, facilitating a common look and feel across both platforms. The new version also supports C and C++ compilers and automatically generates dynamic linked libraries (DLLs).

Gpf Systems Inc.

Phone: (800) 831-0017 or (203) 873-3300

Fax: (203) 873-3302

**Novell NetWare for OS/2.** NetWare for OS/2 is a networking solution that lets companies combine the





advanced services of NetWare 4.01 and OS/2 while realizing significant cost savings through hardware consolidation. With NetWare, a single PC can function as a NetWare 4.01 server, an OS/2 application server, a gateway to a host computer, and a user workstation.

Novell Inc.  
Phone: (801) 429-7000  
Fax: (801) 453-1267

**BocaSoft System Sounds and WipeOut.** System Sounds associates pre-recorded audio with a variety of system, mouse, or keyboard events, and includes a library of pre-recorded sound effects as well as an integrated record mechanism. WipeOut is an

audio/visual screen saver that includes 16 animated displays with integrated digital audio.

BocaSoft Inc.  
Phone: (407) 392-7743.  
Orders: Indelible Blue Inc.  
(800) 776-8284 or (919) 834-7005.  
Fax: (919) 783-8380

**Buzzwords WinGEN for OS/2 2.1.** WinGen for OS/2 creates 32-bit GUI DBMS database applications. The screen designer generates source code for several OS/2 C++ compilers.

Buzzwords International Inc.  
Phone: (314) 334-6317  
Fax: (314) 334-0794

**Sector Seven DEPGEN 2.0.** DEPGEN reads C source code and includes files to create a makefile and dependency list. Powerful new features support multiple directory projects, cross-compilers and linkers, libraries, and DLLs.

Sector Seven  
Phone: (703) 866-9477

**Commix MultiMaster and DisplayMaster.** MultiMaster is a tool for creating multimedia database applications. DisplayMaster is a multimedia browser/organizer.

Commix SP Inc.  
Phone: (703) 356-9858  
Fax: (703) 356-6148



## UTILITIES

**CCT Back in a Flash!** Back in a Flash! is an archive and backup facility for OS/2 2.1. In addition to file compression and automatic backups, Flash! backs up files to diskettes, hard drives, and LAN-attached drives.

CCT Inc.  
Phone: (612) 339-5870  
Fax: (612) 339-5965

**Symantec Norton Commander OS/2.** Peter Norton's latest brainchild is an OS/2 2.x file management utility which features graphical manipulation of FAT and HPFS files.

Symantec Inc.  
Phone: (800) 343-4714  
Fax: (303) 727-4611

**Software Builders ZIPMAN for OS/2.** ZIPMAN is a full-featured GUI interface for PKZIP and PKUNZIP file compression/decompression utilities. It utilizes the Windows multi-document interface feature so that users can open more than one ZIP file at a time for viewing.

Software Builders Inc.  
Phone: (800) 432-0025 or (404) 319-9621

**Rightware LinkRight.** Rightware's LinkRight is a 32-bit parallel and serial port file transfer utility. LinkRight features file copying to and from OS/2 and DOS systems, file compression, CRC checking, and HPFS and long file name support.

Rightware Inc.  
Phone: (301) 762-1151  
Fax: (301) 762-1185



## APPLICATIONS

**CadWare Pj2 CAD System.** Pj2 CAD System is a 32-bit 2D-3D CAD package for OS/2 2.x. Geometrical functions, graphic editing, and associative dimensioning round out CadWare's latest release. Pj2 CAD System uses multi-threading and fast semaphores and is highly customizable.

CadWare, distr. by Indelible Blue Inc.  
Phone: (919) 834-7005  
Fax: (919) 783-8380

**Arcadia Workplace Companion for OS/2.** Workplace Companion is a personal information manager designed for the OS/2 2.1 32-bit Workplace Shell environment. Calendar, clock, appointments, telephone numbers, addresses, to-do lists, and notepad are all included.

Arcadia Technologies Inc.  
Phone: (818) 446-6945  
Fax: (818) 447-4212





## DEVELOPER SUPPORT

### **IBM Developer Connection for OS/2.**

The IBM Developer Connection for OS/2 is a new service designed to provide OS/2 developers with the technical information needed to create OS/2 applications. The Developer Connection will also feature updated information, tips, techniques, sample applications, and beta code for upcoming IBM applications.

The service, which consists of quarterly CDs and newsletters as well as electronically-accessible developer support, costs \$199.95 per year. However, developers who sign up for the Developer Connection before September 30, 1993 are eligible for an introductory price of \$149.95 per year.

For more information about receiving the Developer Connection, call 1-800-633-8266.

**New I.V. League Catalogue.** The IBM OS/2 Independent Vendor (I.V.) League showcases hundreds of books, magazines, courseware, and consulting services that support OS/2. Now, select member products are featured in the all new *I.V. League Product Catalog*. To order a free copy, call 1-800-342-6672.

## OS/2 CALENDAR

|                           |                                                                                           |
|---------------------------|-------------------------------------------------------------------------------------------|
| September 9               | C.A.M.P., Chicago<br>(708) 291-1360                                                       |
| October 5-8               | NetWorld, Dallas<br>(800) 829-3976                                                        |
| October 5-8               | Embedded Systems<br>Conference, Santa Clara, Calif.<br>(415) 905-2220                     |
| October 11-15             | Autodesk University,<br>San Francisco<br>(415) 905-2354                                   |
| October 17-20             | OS/2 Professional Interchange<br>Palm Desert, Calif.<br>(800) 438-6720 or (203) 261-6227  |
| October 18-22             | CASE World, Boston<br>(508) 470-3880                                                      |
| October 18-22             | C++ World, Dallas<br>(212) 274-9135                                                       |
| October 19-21             | PC Expo, Chicago<br>(800) 829-3976                                                        |
| October 26-28             | Common Desktop Environment<br>Developers Conference,<br>San Jose, Calif. (800)225-4698    |
| October 31-<br>November 5 | ColoradOS/2 Developers<br>Conference, Colorado Springs<br>(719) 576-4600 or (800)648-5717 |

## To Appear In Product Watch

If you would like your product to appear in a future Product Watch column, please fax a short press release to:

*OS/2 Developer Product Watch*  
fax (407) 495-4421

Press releases will be edited. Space is limited; *OS/2 Developer* cannot guarantee the appearance of any product in these pages.







Every computer system has resources that must be shared by all applications executing in its environment. This article describes a performance monitoring tool that can help determine whether those resources are being used effectively. **BY LAURA ADAMS**

# SPM/2: Fine-Tuning Your Application



Laura Adams

**O**n any single computer system or network of systems, there are limited resources (such as CPU, RAM, and disk) that need to be shared by all applications executing in that environment. Computer system users, system administrators, and application designers are all interested in how to obtain the most efficient use of these resources and the highest performance at the lowest cost. This means figuring out how to modify parameters, applications, and workloads so the limited resources are used most effectively. To accomplish these tasks, performance data collection and monitoring tools are needed, as well as information on how to interpret the data and where to apply the interpretation in tuning the system.

2.1, small fixes are needed for Theseus2 and the SPM/2 Query function. These fixes can be obtained from OS2TOOLS (SPMFIXES package), OS2BBS (SPMFIXES package under "OS/2 Selective Fixes" in the Software Library), or CompuServe (SPMFI.ZIP file of Library 9 in the OS2DF2 forum).

### SPM/2 2.0 OVERVIEW

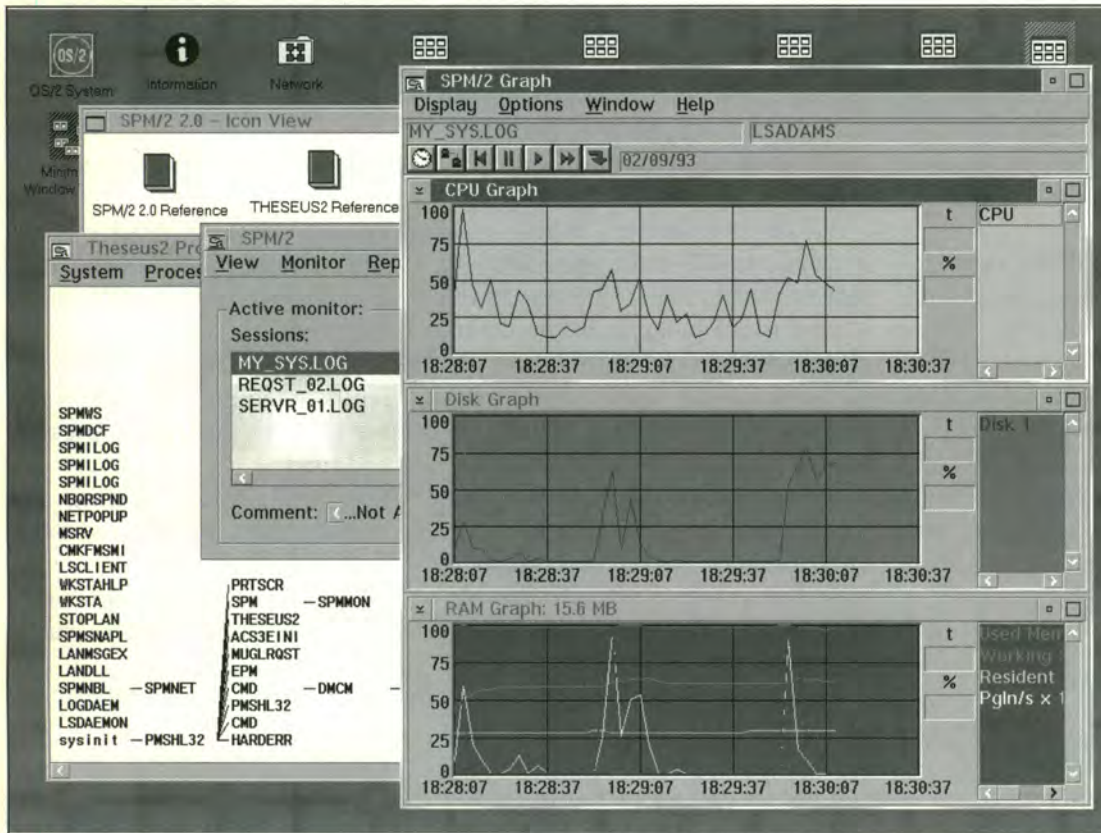
SPM/2 2.0 consists of performance data collecting, recording, graphing, reporting, and analyzing facilities that enable performance management of critical OS/2 2.x system resources, as well as resources of 16- and 32-bit applications executing in an environment of OS/2 2.0 plus Service Pak XR06055 or higher. In addition to collecting performance data locally on a given

*Computer system users, system administrators, and application designers are all interested in how to obtain the most efficient use of resources and the highest performance at the lowest cost.*

This article discusses System Performance Monitor/2 2.0 (SPM/2 2.0), a performance monitoring product for the 32-bit OS/2 environment that helps accomplish performance-related tasks. To run SPM/2 2.0 under OS/2

system, SPM/2 2.0 supports remote monitoring of IBM OS/2 LAN Servers and Requesters via a remote named pipe interface. Monitoring is possible from OS/2 LAN Server running OS/2 2.x or OS/2 LAN Requester 3.0 with Peer Services.





**Figure 1.** SPM/2 Graphing, Control Panel, and Theseus2

SPM/2 2.0 provides the following key features:

- A Presentation Manager-based control panel providing centralized control for performance monitoring activities
- Collection of extensive performance metrics for CPU, memory, files, disk, cache, printer port, and communication port resource performance data
- Collection of user-defined performance metrics from user-written applications
- Discovery capability of LAN-attached SPM/2-enabled workstations
- Support for multiple concurrent monitoring sessions
- Concurrent graphing and recording activity
- Graphical playback of previously recorded data
- Ability to graphically view performance of any workstation within a single monitor session
- Workstation-, application-, process- and thread-level report filters
- Support for processing multiple recorded log files into a single report
- Enhanced function and usability for the memory analyzer, including a Presentation Manager- and Hyperblock-based interface
- API Support enabling user applications and device drivers to register performance metrics for collection
- API Support enabling collection of real-time performance data and processing of historical data
- Online hypertext user guide and references

SPM/2 2.0 is primarily useful in an appli-





cation development or small LAN environment. For application developers, SPM/2's performance information and user-hook capabilities can help produce more efficient applications, help locate defects, and choose smarter application designs. In a small LAN environment, performance data collected by SPM/2 2.0 can help isolate problems quickly and accurately, speeding resolution.

For system administrators responsible for watching the performance of a large number of remote LAN-attached systems, SPM/2 2.0 can be combined with LAN Management Utilities/2 2.0 (LMU/2 2.0). LMU/2 adds the capability to define thresholds on the various resources collected by SPM/2 and generates alerts when a threshold is

**SPM/2 2.0 Control Panel.** The central point of control for an SPM/2 2.0 function is a Presentation Manager-based control panel from which users can define the characteristics of each monitoring session (workstations, resources, collection frequency, and so on), control the starting and stopping of graphing and recording activities, and run reports from recorded data.

While monitoring is active, the control panel lists the monitoring sessions, their status, and the set of workstations being monitored for each session.

**Defining Monitoring Sessions.** Monitoring sessions are controlled by monitor session description, or .LOG, files. These files contain the description of the monitor session, as well as any data recorded during monitoring. The monitor session description includes:

**Workstations**—The user's first specifications when creating a .LOG file are the names of the workstations to be monitored. To aid in identifying the names of workstations on the LAN, SPM/2 provides a query function that asks the network for the names of workstations enabled for SPM/2 2.0 performance data collection. This gives users a list from which to choose. When multiple workstations are monitored from a single .LOG file, the performance data for these workstations is recorded into the same .LOG file.

**Resources**—SPM/2 collects performance data from architected performance metrics, or "hooks," in the OS/2 2.0 operating system. These hooks are simply counter and timer control blocks that count the occurrences of various events and activities in the system. They do not generate messages or event records—

SPM/2 only gets performance data when it specifically asks for it.

Users can select several major resources when defining a monitoring session (with examples of the details collected by SPM/2 in parentheses). These include:

- CPU (execution and interrupt time)
- Physical disk (number of read and write operations, bytes read and written, time spent reading and writing)
- Memory (paging information: faults, page ins and outs, number of pages discarded, present, and free)
- Files (time spent reading and writing files, number of bytes read and written)
- FAT cache (cache hits and misses for reads and writes, cache bypasses)
- HPFS cache (cache buffer information, number of file open and closed)
- Printer port (number of printer write operations and number of bytes written)
- Communications port (number of communications port reads and writes, bytes read and written, time reading and writing, number of software and hardware overrun errors).

In addition to specifying which resources are to be collected, users also specify whether or not to collect application-, process-, and thread-level data.

**Collection and Recording Frequencies**—The collection frequency defines how often SPM/2 reads performance data from the operating system. The recording frequency, which must be a multiple of the collection frequency, defines how often the recording application averages samples read at the

*When multiple workstations are monitored from a single .LOG file, the performance data for them is recorded into the same .LOG file.*

exceeded. These alerts can also be routed to NetView on the host. LMU/2 also stores performance data collected from SPM/2 2.0 in an OS/2 database, allowing an administrator to run reports and look at performance and capacity trends over time.

### **SPM/2 2.0 PRODUCT FUNCTION**

The following sections describe functions provided by SPM/2 2.0. Figure 1 illustrates several of these functions.





Report Type : PROCESS ( SUMMARY )  
Report Date : 09-16-92  
Title : Sample Process Report with CPU, FAT Cache, and Disk  
Time : 18:13:35  
Filename : D:\SPMV2\LOGS\SAMPLE.RDF

Monitor Session : D:\SPMV2\LOGS\CACHE.LOG ( )

Start Date/Time : 09-14-92 18:41:46  
Stop Date/Time : 09-14-92 18:45:13

Sum Interval : 00:03:26.97

\*\*\*\*\*

Workstation : LOCAL

#### CPU

| Summary  | Processor | Threads      | Interrupts | Service       | Application | Program | PID          |
|----------|-----------|--------------|------------|---------------|-------------|---------|--------------|
| Time     | % Util    | Busy Time    | Dispatch   | Avg Timeslice | % Util      | Rate    | Time         |
| 18:45:13 | 67.03     | 00:02:22.100 | 6849       | 00:00:00.021  | 12.05       | 97      | 00:00:17.123 |
|          | 0.66      | 00:00:01.370 | 132        | 00:00:00.010  |             |         | COMM MGR     |
|          | 0.66      | 00:00:01.370 | 127        | 00:00:00.011  |             |         | ACS3EINI 019 |
|          | 0.00      | 00:00:00.000 | 5          | 00:00:00.000  |             |         | CMKFMSMI 017 |
|          | 22.44     | 00:00:46.442 | 1500       | 00:00:00.031  |             |         | SPM/2        |
|          | 16.69     | 00:00:34.550 | 1142       | 00:00:00.030  |             |         | SPM 044      |
|          | 2.98      | 00:00:06.178 | 132        | 00:00:00.047  |             |         | PMDCF 047    |
|          | 2.59      | 00:00:05.354 | 178        | 00:00:00.030  |             |         | SPMILOG 046  |
|          | 5.78      | 00:00:00.360 | 34         | 00:00:00.011  |             |         | SPMISTOP 049 |
|          | 0.00      | 00:00:00.000 | 14         | 00:00:00.000  |             |         | PMSNAPL 009  |
|          | 30.46     | 00:01:03.043 | 4628       | 00:00:00.014  |             |         | -unassigned- |
|          | 0.00      | 00:00:00.001 | 13         | 00:00:00.000  |             |         | LANDLL 003   |
|          | 1.50      | 00:00:03.109 | 639        | 00:00:00.005  |             |         | PMSHELL 007  |
|          | 30.24     | 00:00:42.468 | 2796       | 00:00:00.015  |             |         | PMSHELL 011  |
|          | 22.30     | 00:00:14.402 | 1078       | 00:00:00.013  |             |         | PMSHELL 048  |
|          | 0.02      | 00:00:00.005 | 46         | 00:00:00.000  |             |         | HARDERR 010  |
|          | 2.18      | 00:00:03.059 | 56         | 00:00:00.055  |             |         | EPM 028      |

#### FAT CACHE

| Hit % | Hits | Misses | Bypasses | Cache Read | Cache Write | Forces |
|-------|------|--------|----------|------------|-------------|--------|
|       |      |        |          | Hits       | Misses      |        |
| 79.82 | 4866 | 1230   | 0        | 4819       | 1198        | 183    |

#### DISK

| Physical ID | % Util | Access Rate | Requests | Sectors | Read Avg Sectors | Read Avg Time | Requests | Sectors | Write Avg Sectors | Write Avg Time |
|-------------|--------|-------------|----------|---------|------------------|---------------|----------|---------|-------------------|----------------|
| Disk 01     | 49.03  | 13          | 2401     | 15.1K   | 6                | 00:00:00.033  | 422      | 5835    | 13                | 00:00:00.057   |

Figure 2. SPM/2 2.0 process-level report



Working Set for the entire System: Use the "Functions" pull-down to start and stop data collection.

Collection started: interval = 5, Working Set is 12 intervals.

| current time |      | Total |          |      | Process |             |       |
|--------------|------|-------|----------|------|---------|-------------|-------|
| hh:mm:ss.tt  | now  | ws    | accessed | Free | Idle    | contributed | total |
| 18:39:40.91  | 1071 | 1071  | 1071     | 1    | 44      | 9           | 18    |
| 18:39:45.78  | 989  | 1160  | 1160     | 1    | 71      | 8           | 18    |
| 18:39:51.35  | 1129 | 1288  | 1288     | 0    | 61      | 8           | 18    |
| 18:39:55.85  | 1169 | 1401  | 1401     | 4    | 46      | 9           | 18    |
| 18:40:00.78  | 917  | 1455  | 1455     | 0    | 59      | 9           | 18    |
| 18:40:05.78  | 947  | 1472  | 1472     | 0    | 60      | 9           | 18    |
| 18:40:10.78  | 919  | 1472  | 1472     | 0    | 59      | 9           | 18    |
| 18:40:15.78  | 896  | 1472  | 1472     | 0    | 59      | 9           | 18    |
| 18:40:20.85  | 933  | 1474  | 1474     | 0    | 57      | 9           | 18    |
| 18:40:25.78  | 1216 | 1497  | 1497     | 0    | 60      | 9           | 18    |

| current time |      | Total |          |      | Process |             |       |
|--------------|------|-------|----------|------|---------|-------------|-------|
| hh:mm:ss.tt  | now  | ws    | accessed | Free | Idle    | contributed | total |
| 18:40:30.78  | 1027 | 1501  | 1501     | 0    | 57      | 9           | 18    |
| 18:40:36.94  | 984  | 1547  | 1547     | 0    | 85      | 10          | 18    |
| 18:40:40.75  | 979  | 1517  | 1548     | 8    | 83      | 9           | 18    |
| 18:40:46.25  | 1127 | 1492  | 1550     | 8    | 81      | 9           | 18    |
| 18:40:50.82  | 847  | 1439  | 1550     | 8    | 81      | 9           | 18    |
| 18:40:55.85  | 1189 | 1464  | 1585     | 0    | 59      | 9           | 18    |
| 18:41:00.82  | 1019 | 1466  | 1586     | 0    | 51      | 9           | 18    |

17 samples collected.      Total  
Min Working Set:      1071  
Max Working Set:      1547  
Avg Working Set:      1422.82  
Standard deviation:      129.163

Absolute minimum number of pages:      1216  
Recommended number of pages:      1547  
Total number of accessed pages:      1586

**Figure 3.** Example of system working set

collection frequency and stores them in the .LOG file.

Although SPM/2 has optimized its overhead, two things affect intrusiveness during data collection. First, specifying collection of application-, process-, or thread-level data will result in more counter control blocks (one for every process and thread in the system) and hence more data will be read. Also, the collection frequency will determine how often data is

read. Specifying a smaller collection frequency will, of course, result in more data being read.

In both situations, reading more data results in larger .LOG files when recording and, in a remote environment, more LAN traffic. You can work with these two parameters to achieve a balance acceptable for your monitoring situation.

**Graphing and Recording.** Monitoring is

started by selecting the desired .LOG file and then choosing Graphing, Recording, or Both. SPM/2 2.0 allows you to graph and record at the same time. The status window of the SPM/2 control panel will remind you which functions are active for a particular .LOG file.

The graphing facility provides a graphical presentation of utilization on a system-wide basis for the following resources:



- CPU
- Physical disk (up to 24 physical drives, with support for SCSI, ESDI, and ST-506 device drivers)
- Memory utilization, which includes the total physical memory (the value displayed in the title bar), used memory, working set memory (the number of pages touched during a specified period of time), resident (nondiscardable and nonswappable) memory, pages swapped in, and pages swapped out.

To obtain detailed process- and thread-level information or data about other system resources, the recording and report functions must be used.

Although your .LOG file may be defined to monitor multiple workstations, the graphing facility displays the activity for only one workstation at a time. However, by clicking on the Node icon in the graphing window, you will be presented with a list of the workstations defined in the active .LOG file and you can change which workstation is being graphed. A status bar in the graph window tells you the .LOG file name and specific node currently being graphed.

Usability features of the graphing facility include:

- A legend that defines what each line in the graph represents
- The current time displayed across the X-axis of each graph window
- A cross-hair cursor that can be positioned over any point on a graph line and clicked to display the exact X/Y value (time and percent utilization) of that point
- Ability to select graph colors.

The graphing facility also has a playback function that allows you to review previously recorded data. If your monitor session was started

with simultaneous graphing and recording, you can switch over to playback mode at anytime and review the data already recorded during that session. Current performance data will continue to be recorded to the .LOG file while in playback mode.

**Running a Report.** Before running a report, users create a report description (.RDF) file that defines the .LOG file(s), resources, summarization level (workstation, process, or thread), and format to be used in creating that report. An administrator could have a standard set of monitoring session descriptions to be run on a regular basis, with corresponding .RDF files for generating periodic summary reports.

Multiple .LOG files can be combined into one report, although it is

from the operating system and produces more meaningful utilization information. Examples of information calculated by the report facility are:

- Percentage of CPU utilized by process and thread
- Average physical disk read and write times
- Page-in and page-out rates
- Paging disk utilization
- Cache hit and miss ratios
- Average file read and write times
- Printer and communication port throughput rates

To locate a performance problem using the report facility, you can specify a subset of resources to analyze, a subset of the time period to analyze, and the report summariza-

*To locate a performance problem with the report facility, you can specify a subset of resources to analyze, a subset of the time period to analyze, and the report summarization level, which defines how detailed the report is.*

advised that the same resources be collected in each .LOG file. Combining .LOG files might be useful if new workstations are added to the network at a time when it is inconvenient to add these workstations to the existing .LOG file (mid-week, for example). By defining a new .LOG file for the new workstations, data could be collected from those workstations and then folded into the weekly report.

In generating a report, the SPM/2 report facility performs calculations on the raw data collected

tion level, which defines how detailed the report is. The summarization levels supported (from least to most detail) are workstation, application, process, and thread.

Specifying any of the more detailed levels will include all the lesser-detailed levels of detail in the report. For example, a process report will include summary lines for each application and workstation. Figure 2 shows an example of a process-level report.

If an application summarization







level is specified, SPM/2 will group processes from the same application under a common application name and include a resource summary line for that application. Several OS/2 system applications (such as LAN Server and Communications Manager) come predefined with SPM/2. Additionally, SPM/2 provides the capability of defining

*SPM/2 APIs enable application and device-driver developers to define and register application-specific counters and timers that can help optimize code performance.*

your own applications as any set of executable modules.

The supported report formats are summary, tabular, delimited (spreadsheet compatible), and dump (raw performance data values).

**Memory Analyzer: Theseus2.** The SPM/2 Memory Analyzer (Theseus2) provides application developers with insight into OS/2 memory management. An important feature of Theseus2 is its capability to provide working set information (which gives the amount of memory actually referenced over a given time interval). Working set information can help you more accurately determine the amount of physical memory needed in your system to minimize swapping. Figure 3 shows an example of a system

working set report for an interval of about 1.5 minutes.

Some important usability features of Theseus2:

- A main window that displays all processes currently active in the system. This is useful for knowing whether or not background processes are still active.
- A Presentation Manager interface into OS/2 memory management information
- Hyperblock linking that allows navigation from one memory control block to another by simply double-clicking (with the mouse) on the address of the control block to be viewed
- Ability to output reports to the printer or a file and highlighted text to the printer or the clipboard
- Extensive help that explains every field of the Theseus2 windows
- On-line hypertext documentation that provides valuable information about OS/2 2.x memory management concepts.

Examples of the memory information provided (which can be displayed as linear, virtual, or physical addresses) are:

- System information—loaded device drivers and modules, a global descriptor table, system anchor segment, system memory usage, system arena and page tables, free, idle, or locked memory, memory utilization, and working set for the entire system.
- Process Information—per task data area, local descriptor table, page table, private and shared arena tables, memory utilization, and working set by individual process.
- Register information—processor

control registers, current interrupt descriptor table, and current task state segment.

**Directory Analyzer.** The SPM/2 directory analyzer provides the capability to analyze disk capacity (data file and size) information and report the information by directory, subdirectory, file, and cluster level. The specific information provided includes the data size of any file in bytes (how much data is in the file), the file allocation size (the number of clusters on a disk), and the count of files on the disk.

**Application Programming Support.** SPM/2 supports APIs of two types. One type enables line-of-business applications to process performance data in real time, allowing an application to initiate and terminate monitoring sessions, open and close .LOG files, and read data from those files. These applications can then utilize the data in ways specific to the particular business.

Other SPM/2 APIs enable application and device driver developers to define and register application-specific counters and timers that can help optimize the performance of their code. Once registered with the operating system, SPM/2 will see these user-defined metrics and read them along with the OS/2 2.0 resource metrics. The values of these user-defined metrics can be formatted and viewed using the Report Facility's Dump format.

**Command Line Interface.** Commands are provided to start and stop the data collection functions, graphing, recording, as well as run reports. This command-line interface enables SPM/2 facilities to be remotely executed from a NetView-based workstation using the IBM Remote Operations or IBM Distributed Console Access Facility programs.



## USING THE SPM/2 2.0 DATA: AN APPROACH

With these functions, users can narrow in on a performance problem by first running the graphing facility to see which resources (CPU, disk, or RAM) are being used. Users can also choose to start recording simultaneously with graphing, then generate a workstation-level report to see information not displayed in the graph (such as paging information or number of bytes read and written to the disk).

Depending on which resource appears to be a problem, either more data should be collected or Theseus2 should be used. For any resource problem other than memory, modify the .LOG file description to specify collection of additional related resources (for example, if disk is a problem, ask for File information to see which files are being used heavily). You will probably want to include application-, process-, and thread-level detail to generate more comprehensive reports.

Rerun the scenario of interest with this modified .LOG file, then run a summary report for the entire scenario and see which processes or files are involved in the problem. Also try playing back the .LOG file, pausing at locations of peak activity. With the cross-hair cursor, determine times that surround the period of peak activity. Using these times, run other reports, isolating just this time period. Then focus on the processes using the most resources, asking what these processes might be doing and how things might be changed.

If the resource of concern is memory, including heavy swapping activity, the Theseus2 memory analyzer should be used to determine how much memory each process is using, as well as the level of swap activity.

## SCENARIOS

**Scenario 1.** A particular company has a Database Manager application running at several remote sites, each with its own database server. The headquarters location has been getting complaints that things are slowing down during certain periods each day. This information is insufficient to determine the problem.

To narrow in on the problem, remote SPM/2 2.0 monitoring sessions are initiated to run over the time during which complaints are received. An initial look at the reports indicates that server activity changes from relatively quiet to quite congested in a short period of time (as determined by the number of new processes that appear in the report). Further monitoring and use of Theseus2 reveals that swapping activity also increases during this time.

To obtain more information, employees are asked what they typically do during these periods. It is discovered that after initially accessing the database in the morning, people drift into other work. Later, after a midmorning break, they return to their workplaces and start initiating transactions against the database, causing many applications previously swapped out due to inactivity to be swapped back into the system and run concurrently. It is decided to add more RAM to the servers, as well as define some employee work procedures, to alleviate the situation.

**Scenario 2.** A certain business application is found to be getting slower and slower. SPM/2 2.0 shows that the most heavily used resource is the physical disk. When SPM/2 is run again, this time requesting File information, reports show that several data and logging files owned by the application are continually

accessed. In addition, application code is loaded quite frequently and is found to reside on the same disk as the data and logging files.

More detailed analysis is performed (by changing collection frequency and report intervals) to determine exactly when the various files and program modules are accessed. As a result, a new physical disk and disk controller is added to the system. Files and programs are moved to this new disk so that when they are accessed during the same period they are not on the same disk.

*Depending on which resource appears to be a problem, either more data should be collected or Theseus2 should be used.*

**Scenario 3.** During testing of an application under development, it is found that the system almost grinds to a halt when certain common tasks are performed. In some cases, the keyboard stops responding completely. SPM/2 is brought in, data is recorded to a .LOG file, and a process- and thread-level report is run. The CPU section gets immediate attention because a few application processes monopolize the CPU while others appear unable to get any processing time, even though it would be expected that they get at least a small amount of the CPU.

Theseus2 is run to find the priority of each active process. It is dis-







covered that several of the application processes have set their own priorities to time-critical, rather than relying on the OS/2 scheduler to share the CPU appropriately among the processes. Further questioning also reveals that **PRIORITY** has been set to **ABSOLUTE** in config.sys. (Setting **PRIORITY** to **ABSOLUTE** keeps some of the OS/2 components, such as Communications Manager, from being able to dynamically modify the priorities of certain time-dependent pieces of code.)

The developers are asked to let the operating system set the priorities of their processes. If needed, minor adjustments can be made later (but nothing so drastic as setting **PRIORITY** to time-critical).

**Scenario 4.** The developer of an application wants to optimize the application's performance as much as possible. One of the resources that needs optimizing is the internal buffers, which are allocated and released as needed. The developer defines application-specific counters and timers to see how often these buffers are allocated and to determine the maximum number needed during certain critical periods of the application.

These metrics are registered with the operating system with SPM/2 APIs. The developer then has the application update the counters whenever the buffers are allocated

and released. With this information, the developer can determine the optimal size of each buffer so buffers aren't allocated too often (which causes excess processing overhead), and the buffer space isn't too large (which results in excess unused memory being tied up).

### CONCLUSION

Performance measurement and analysis is required at every stage in the lifecycle of a LAN system, including planning, design, implementation, use, and maintenance. This type of analysis can show how well a LAN system is performing and whether tuning is required.

SPM/2 and LMU/2 provide capabilities for collecting, monitoring, logging, reporting, and analyzing system performance data. Future products will expand on this base of information, enabling more control from a centralized location.

**Laura Adams**, IBM Corp., 11400 Burnet Rd., Austin, Texas 78758. Adams is an advisory programmer in the LAN systems area of IBM's Personal Systems Products division. She joined IBM in 1976 and in 1986 joined the OS/2 Database Manager development team. She is currently working in market support and planning for SPM/2 and LAN NetView Monitor. Adams holds a B.A. in mathematics from Hope College.

### REFERENCES

For more information on SPM/2, questions can be asked in the OS2SPM20 CFORUM. IBM employees can access this forum on the IBM PC disk; customers can access it on the OS2BBS.

Readers can also check the IBM OS2 Forum, OS2DF2, on CompuServe.







Performance enhancements, simple to complex, consume a great deal of time. The link step, however, is often overlooked in the quest for a smaller, faster application. This article describes LINK386 options that relate to performance. **BY ALLEN WYNN**

# Using LINK386 Options To Improve Performance

**P**erformance enhancements, simple to complex, consume a great deal of time. The link step, however, is often overlooked in the quest for a smaller, faster application. This article describes LINK386 options that relate to performance. LINK 386, a utility that ships with OS/2 2.x and the OS/2 Toolkit, can help improve speed and reduce size in your applications.

## **/ALIGNMENT**

The **/ALIGNMENT** option specifies the alignment factor for an executable. The alignment factor is used to determine the starting offset of all pages within the executable file. The parameter must be a power of 2 and must be in the range of 2 to 32,768. All pages in the executable will be based on a multiple of *n* bytes from the beginning of the file.

Alignment factors greater than 4,096 will waste disk space because pages in OS/2 are 4,096 bytes in length. An alignment factor of 2 will produce the smallest executable.

The default alignment factor of 512 will produce an executable with pages that are sector aligned. This may produce an executable that loads marginally faster. The file will usually be much larger, and the marginal load performance does not usually offset the additional disk space requirements.

## **/BASE**

The **/BASE** option specifies a preferred load address. The preferred base address is rounded up to the nearest multiple of 64K. The first object is assigned the preferred load address. Each additional object

is assigned the next available multiple of 64K. LINK386 will then pre-apply all internal relocation records based on the preferred load address.

All .EXE files are loaded at a base address of 64K, so the preferred load address for a .EXE must be 64K. If another preferred load address is specified for an .EXE, a warning message will be displayed, and the preferred load address will be assigned to 64K. Because the .EXE file is guaranteed to be loaded at its preferred load address, LINK386 will not write the internal relocation records to the target file. This results in a smaller file because there are no internal relocation records. The executable will load faster because the internal fixes do not have to be applied while the program is loading.

A dynamic link library (DLL) can be based as well; however it is not recommended. The actual



Allen Wynn

*If another module has been loaded in a DLL's preferred load address, the loader will need to reapply the internal relocation records.*

load address cannot be determined until run time. If another module has been loaded in a DLL's preferred load address, the loader will need to reapply the internal relocation records. LINK386 must write the internal relocation records of the DLL to





the target file, so they will be available to the loader. Basing a DLL will not result in a smaller file. If the loader has to reapply the internal relocation records of a DLL, the DLL will actually load more slowly than if the DLL was not based.

The preferred load address for an .EXE must be 0x10,000 (64K). The preferred load address for a DLL should be a large number less than 0x18000000.

### **/DEBUG or /CODEVIEW**

The /DEBUG option directs LINK386 to look for symbolic data and line number information in object modules and include this information in the executable file. The /CODEVIEW option produces exactly the same results as the /DEBUG option, regardless of the symbolic debug format (for example, AIX style, IBM PM Debugger, and so on).

While the /DEBUG option is a popular debugging tool, symbolic debugging information should only rarely be

```
NOP
PUSH    CS
CALL    NEAR function
```

**Figure 1.** /FARCALLTRANSLATION replacement code for CALL FAR

```
JMP     NEAR function
NOP
NOP
```

**Figure 2.** /FARCALLTRANSLATION replacement code for JMP FAR

to compress pages within the executable. These pages are automatically expanded when the program is executed.

The current /EXEPACK algorithm compresses patterns of repeated bytes in pages of data. Pages of code do not usually have patterns of repeating bytes long enough to make packing worthwhile.

The /EXEPACK option will not always create a smaller executable. However,

/EXEPACK flag will usually cause your program to load faster. The amount of time to read a section of compressed data from disk and decompress the data in memory is usually less than the time to read an uncompressed page from disk.

### **/FARCALLTRANSLATION**

The /FARCALLTRANSLATION option causes LINK386 to optimize far calls and far jumps to the same segment. The CALL FAR instruction is replaced with the code shown in Figure 1, and the JMP FAR instruction is replaced with the code shown in Figure 2.

These new instruction sequences eliminate loading a segment register on the call or jump, which results in faster execution when running in protect mode. A load-time relocation is eliminated, resulting in a smaller file that loads faster.

The /PACKCODE option is often used in conjunction with the /FARCALLTRANSLATION option. The /PACKCODE option can be used to combine code segments, potentially increasing the number of CALL FAR and JMP FAR instructions made to a target address in the same segment. The /FARCALLTRANSLATION option does not affect near calls.

### **/PACKCODE**

The /PACKCODE option does *not* compress code. The /PACKCODE option combines adjacent code segments. Adjacent code segments receive the same segment address, and offsets are adjusted as required.

*To achieve maximum compression with the /EXEPACK flag, all data items that have the same initial value should be in the same area of the data segments. Any data items that are not initialized to a nonzero value can be initialized to zero.*

included in executables that are shipped as part of a final product.

### **/EXEPACK**

The /EXEPACK option is probably the most ignored LINK386 option, however, it can produce significantly smaller executables. These smaller executables usually load faster and perform better under low-memory, high-swapping situations.

The /EXEPACK option causes LINK386

an executable linked with LINK386 and the /EXEPACK option will not be larger than the same executable linked with LINK386 without the /EXEPACK option.

To achieve maximum compression with the /EXEPACK flag, all data items that have the same initial value should be in the same area of the data segments. Any data items that are not initialized to a nonzero value can be initialized to zero.

In addition to saving disk space, the



## ADVERTISER INDEX

| Company, Fax Number                        | Page  | Company, Fax Number                  | Page    |
|--------------------------------------------|-------|--------------------------------------|---------|
| Borland International . . . . .            | COV4  | Intelligent Environments . . . . .   | 44      |
| (408) 439-0115                             |       | (508) 640-1090                       |         |
| Burton Systems Software . . . . .          | 27    | Intersolv . . . . .                  | 36      |
| (919) 233-0716                             |       | (503) 645-4576                       |         |
| CCT . . . . .                              | 55    | Kaseworks. . . . .                   | 11      |
| (612) 339-5965                             |       | (404) 448-4163                       |         |
| Cognitive Software Inc. . . . .            | 55    | MSR Development . . . . .            | 79      |
| Creative Systems Programming . . . . .     | 65    | (409) 560-5964                       |         |
| (609) 234-1920                             |       | Magus . . . . .                      | 43      |
| dSoft Development. . . . .                 | 27    | (415) 940-1238                       |         |
| (405) 360-3045                             |       | Micro Focus. . . . .                 | 2       |
| Digitalk Inc. . . . .                      | 58    | (415) 856-6134                       |         |
| (310)645-1306                              |       | Microformatic. . . . .               | 69      |
| Essex Systems Inc. . . . .                 | 55    | (203) 648-9587                       |         |
| (508) 750-4699                             |       | Microway . . . . .                   | 43      |
| Gpf Systems Inc. . . . .                   | 73    | (508) 746-4678                       |         |
| (203) 873-2171                             |       | Mitnor Software . . . . .            | 77      |
| Gamma Tech . . . . .                       | 93    | (918) 357-2869                       |         |
| (405) 359-7391                             |       | Parallel Storage Solutions . . . . . | 81      |
| Graphical Software Interfaces. . . . .     | 27    | (914) 347-4646                       |         |
| (404) 382-6374                             |       | Programmer's Paradise . . . . .      | 4, 16A  |
| Hilbert Computing. . . . .                 | 55    | Real Time Technologies . . . . .     | 55      |
| (913) 829-2450                             |       | (708) 446-6886                       |         |
| HockWare . . . . .                         | 18    | Rimstar Technology . . . . .         | 89      |
| (919) 380-0757                             |       | (603) 778-2408                       |         |
| IBM Canada. . . . .                        | 23    | Sequiter . . . . .                   | COV3    |
| (800) 445-2426                             |       | (403) 448-0315                       |         |
| IBM (DB22) Programming Systems . . . . .   | 32    | Soft & GUI Inc. . . . .              | 9       |
| (800) 445-2426                             |       | (718) 934-2133                       |         |
| IBM GIK. . . . .                           | 13    | Softbridge . . . . .                 | 45      |
| 011-43-1-21145-4490                        |       | (617) 864-7747                       |         |
| IBM Personal Software Products . . . . .   | 96    | Token Technology Inc. . . . .        | 33      |
| IBM Personal Software Products . . . . .   | 14-15 | (415) 965-8658                       |         |
| (407) 982-6408                             |       | Tritus Inc. . . . .                  | 41      |
| IBM Programming Systems Cary Labs. . . . . | 67    | (512) 794-3833                       |         |
| (919) 469-7423                             |       | Van Nostrand Reinhold . . . . .      | 24, 61  |
| IBM Santa Theresa Laboratories . . . . .   | 49    | (606) 525-7778                       |         |
| Impact Software . . . . .                  | 55    | Watcom C for IBM PCs. . . . .        | COV2, 1 |
| (818) 879-5593                             |       | (519) 747-4971                       |         |
| Information Builders Inc. . . . .          | 53    | XVT Software . . . . .               | 28      |
| (212) 629-8819                             |       | (303) 443-0969                       |         |
|                                            |       | Zinc Software Inc. . . . .           | 35      |
|                                            |       | (801) 785-8996                       |         |





If a number is specified, it defines the maximum size of a code segment grouped by the `/PACKCODE` option. If no number is specified, the maximum size will be 65,530. LINK386 will stop adding segments to a group when the next segment will cause the size to exceed the specified size.

LINK386 will pack code segments by default; this option is provided to override a `/NOPACKCODE` option specified in the LINK386 environment variable.

### **`/PACKDATA`**

The `/PACKDATA` option does *not* compress data. The `/PACKDATA` option combines adjacent data segments. Adjacent data segments receive the same segment address, and offsets are adjusted as required.

If a number is specified, it defines the maximum size of a data segment grouped by the `/PACKDATA` option. If no number is specified, the maximum size will be 65,530. LINK386 will stop adding segments to a group when the next segment will cause the size to exceed the specified size.

### **`/RUNFROMVDM`**

LINK386 inserts a DOS stub or code segment, which is loaded and run when an application is executed in DOS, the DOS Box of OS/2 1.x, or in a Virtual DOS Machine in OS/2 2.x.

The `/RUNFROMVDM` option is new for OS/2 2.1 and causes LINK386 to use an alternate DOS stub. This alternate DOS stub will print a message and exit if the program is executed in DOS, the DOS Box of OS/2 1.x, or a Virtual DOS Machine in OS/2 2.0. If the program is executed in a Virtual DOS Machine in OS/2 2.1, the system will cause the program to be started in protect mode.

The `/RUNFROMVDM` option does not cre-

```
SET LINK386=/EXEPACK /FARCALL /ALIGN:2  
  
LINK386 program1.obj,program1.exe;  
  
LINK386 program2.obj,program2.exe /STACK:0x2000;  
  
LINK386 program3.obj /ALIGN:4,program3.exe,/RUNFROMVDM;
```

**Figure 3.** LINK386 Environment Variable example

## **REFERENCES**

DOS 5.00 Technical Reference (IBM Doc. 92F3118)

OS/2 2.0 Toolkit Online Reference (IBM Doc. 10G3355)

ate a smaller executable or an executable that will load faster if started from protect mode. However, if the user tries to execute the program from a Virtual DOS Machine in OS/2 2.1, the user will not be required to switch to a protect mode session and retype the command.

### **`/STACK`**

The `/STACK` option specifies the stack size, in bytes, for the program. Any positive value can be specified. If a program uses little stack space, decreasing the stack size will conserve memory.

## **LINK386 ENVIRONMENT VARIABLE**

The LINK386 environment variable specifies LINK386 options that are to be used each time LINK386 is invoked. The LINK386 environment variable is processed before the command line options.

In Figure 3, the LINK386 environment variable is set to include the `/EXEPACK`, `/FARCALLTRANSLATION`, and `/ALIGN-`

`MENT:2` LINK386 options. The `/ALIGNMENT` option has been abbreviated to `/ALIGN` and the `/FARCALLTRANSLATION` option has been abbreviated to `/FARCALL`.

In this example, `program1` is linked using the `/EXEPACK`, `/FARCALLTRANSLATION`, and `/ALIGNMENT:2` LINK386 options. `Program2` is linked using the `/EXEPACK`, `/FARCALLTRANSLATION`, `/ALIGNMENT:2`, and `/STACK-SIZE:0x2000` LINK386 options. `Program3` is linked using the `/EXEPACK`, `/FARCALLTRANSLATION`, `/ALIGNMENT:4`, and `/RUNFROMVDM` LINK386 options.

**Allen Wynn**, IBM Corp., 1000 N.W. 51st St., Boca Raton, Fla. 33432. Wynn is a senior associate programmer in OS/2 development. Since joining IBM in 1987, he has worked on OS/2 System Test, OS/2 device drivers and MVDM, and OS/2 kernel and file systems. His current assignment is in the OS/2 system performance department. He received a B.S. in information and computer science from the Georgia Institute of Technology.



New for OS/2

# CodeBase 5.0

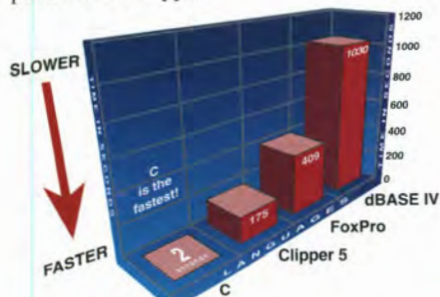
## Discover why FoxPro, Clipper, and dBASE were all written in C.

**T**here is a good reason why your database language was developed in C. In fact, there are many good reasons.

C code is small. C code is fast. C code is portable. C code is flexible. C is the language of choice for today's professional developer. With the growing complexity of database applications, C is a realistic alternative. Now with CodeBase 5.0, you can have all the functionality, simplicity and power of traditional database languages together with the benefits of C/C++.

### C speed - fast code, true executables...

FoxPro, Clipper, and dBASE were written in C primarily for speed. But those compilers don't really compile, they combine imbedded language interpreters into your .EXE. Now that's slow. For dazzling performance you need the true executables of C. With CodeBase you get the real thing, C code. Consider the following statistics, from the publisher of Clipper:



"Sieve of Erasthothenes"  
Benchmark for Prime Number Generation  
Shows C to be incredibly faster!

### C size - small executables, no added overhead...

FoxPro, Clipper and dBASE would like you to believe you need their entire development system to build database applications. But

remember, those products are all written in C. So why do you need to lug all their extra code around? You don't. CodeBase is a complete DBMS, in C. No fat executables stuffed with unused code. No runtime modules. No royalties. Just quality C code. CodeBase is just what you need.

### C portability - ANSI C/C++ on every hardware platform...

No other language exists on more platforms than C/C++. Why rewrite your entire application for DOS, Windows, Windows NT, OS/2 or UNIX? With CodeBase the complete C source code is included, so you can port to any platform with an ANSI C or C++ compiler. Now and in the future.

### dBASE Compatible data, index and memo files...

You want the industry standard. You need compatibility. Sure, dBASE is the standard, but every dBASE compatible DBMS product uses its own unique index and memo file formats. Only CodeBase has them all: FoxPro (.cdx), Clipper (.ntx), dBASE IV (.mdx) and dBASE III (.ndx). Now it's your choice, we're compatible with you.

## Announcing CodeBase 5.0

The power of a complete DBMS, the benefits of C

### NEW - Multi-user sharing with FoxPro, Clipper and dBASE...

Now your multi-user C/C++ programs can share data, index and memo files at the same time as concurrently running FoxPro, Clipper and dBASE programs. No incompatibilities. No waiting.

### NEW - Queries & Relations 1000 times faster...

CodeBase 5.0 now lets you query related

data files with any logical dBASE expression. Our new Bit Optimization Technology (similar to FoxPro's Rushmore technology) uses index files to return a query on a 1/2 million record data file in just a second. Automatically take advantage of this query performance by using our new CodeReporter:

The screenshot shows the CodeReporter application window. The menu bar includes File, Align, Database, Groups, Global, Print, Query, Styles, and Help. The title bar reads 'Code Reporter: "report.rpt"'. The main window displays two reports for 'Product Sales Summary'.

**Report 1: Nov, 1992**

| Month of:              | Nov, 1992  |                    |
|------------------------|------------|--------------------|
| Product                | Quantity   | Value              |
| Database               | 83         | \$25,137.00        |
| Spread Sheet           | 88         | \$21,886.00        |
| <b>Monthly Summary</b> | <b>121</b> | <b>\$47,023.00</b> |

**Report 2: Dec, 1992**

| Month of:              | Dec, 1992  |                    |
|------------------------|------------|--------------------|
| Product                | Quantity   | Value              |
| Database               | 82         | \$24,862.00        |
| Spread Sheet           | 83         | \$18,876.00        |
| <b>Monthly Summary</b> | <b>115</b> | <b>\$43,738.00</b> |

**Summary**

|                |            |                    |
|----------------|------------|--------------------|
| <b>Summary</b> | <b>236</b> | <b>\$91,761.00</b> |
|----------------|------------|--------------------|

To use CodeReporter, simply draw your report, then include it in any program you write. Call 403/437-2410 now for your FREE working model of CodeReporter.

### New - Design complex reports in just minutes...

Our new CodeReporter takes the painstaking work out of reports. Now simply design and draw reports interactively under Windows 3.1, then print or display them from any DOS, Windows or UNIX application.

### SPECIAL - FREE CodeReporter

Order CodeBase 5 before October 31, 1993 and receive CodeReporter for free! This offer includes our no-risk, 90-day money back guarantee, so order today!



**CodeBase 5.0**  
The C/C++ Library for DataBase Management

**Call Now**  
**403-437-2410**

**SEQUITER SOFTWARE INC.** FAX 403-436-2999  
Europe 33.20.24.20.14  
#209,9644-54 AVE., EDMONTON, AB, CANADA T6E-5V1



# Borland Languages:

## Why it's smart to go with the standard



Look for this mark on third-party software products that enhance Borland tools.

### The tools of choice for Windows, DOS, and OS/2

Use Borland tools and you have more than the world's best languages. *Much more.* Borland® Pascal and Borland® C++ for Windows, DOS, or OS/2® give you access to the largest resource of libraries, custom controls, and other add-ins. Whether your projects are

### All the add-ins you need

- Database access
- SQL libraries
- CASE tools
- Multimedia
- Books, classes, and video training
- Communications
- User interface libraries for DOS, Windows, and OS/2
- Graphics and image processing
- Math, scientific, and engineering libraries
- Source control
- Embedded systems tools
- Object-oriented design tools
- Utilities
- Testing automation

in client/server architecture or in embedded systems, Borland has forged partnerships with third-party developers to help you get the job done fast.

### To set standards, you must meet standards

When you use Borland tools, it's easy to understand why they attract the volume of support that makes them the standard. For example,

Borland C++ is the *only* popular compiler that meets the certified ANSI C and AT&T C++ standards. That means you can use essential new features like templates and exception handling\* now.

Whether you're a professional programmer or just starting out, Borland has the innovative tools to get your job done fast. And with third-party support from companies like Phar Lap, Greenleaf, Inmark, KASE Works, Nu-Mega, POET, Rational, Turbo-Power, XVT, and Zinc, it's smarter than ever to go with the standard. Go with Borland Pascal and Borland C++ today!



90-day, money-back guarantee!

**See your dealer or call now,**

**1-800-336-6464, ext. 7075**

In Canada call, 1-800-461-3327.

**Borland is the market leader**

**67%**

**Worldwide languages market share†**

**Borland**

Power made easy  
6362-0001-19

